Sistema de Controle de Acesso

Relatório Final

Lécio C. B. Alves

Departamento de Ciência da Computação Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

Projeto Orientado em Computação II
Orientador: Daniel Fernandes Macedo

charles-L@ufmg.br

1. Introdução

Uma maneira prática e segura de fazer um controle de acesso a uma sala é através de fechaduras eletrônicas com registro e controle de acesso.

A fechadura eletrônica simplifica o processo de concessão de acessos, pois estes podem ser feitos através de um sistema, sem a necessidade de cópias físicas de chaves. A permissão de acesso é dada automaticamente, ao indicar no sistema que determinado usuário pode entrar em determinada sala. Um usuário que deseja passar por uma fechadura eletrônica deve apresentar a sua chave eletrônica, que pode ser, por exemplo, um cartão com dados a serem lidos pela fechadura. O sistema da fechadura consulta um servidor e dependendo da resposta pode conceder ou não o acesso à sala. Com isso o acesso às salas torna-se intransferível, pois ele é atrelado à uma identidade. Este trabalho para ser implantado na UFMG, mas para fins de clareza, será considerado a sua implantação em uma organização genérica.

O objetivo principal deste trabalho é desenvolver um sistema de fechadura eletrônica com controle e registro de acesso de usuários e apresentar o protótipo ao final.

2. Modelagem

Foi feita uma modelagem abstrata usando a linguagem *Alloy*[1]. Esta linguagem não gera código executável e não é apropriada para a implementação, ela permite a declaração de elementos e suas propriedades, expressadas usando lógica de primeira ordem (lógica booleana, implicações) e faz um processo de verificação onde são geradas instâncias aleatórias e válidas dentre todas as instâncias possíveis, de acordo com o modelo e restrições declaradas. Estas instâncias possuem um número limitado de objetos, chamamos este conjunto limitado de *escopo*. Esta limitação de escopo faz do *Alloy* uma linguagem que não é Turing Completa, isto é, não é possível expressar laços de tamanhos arbitrários, por exemplo.

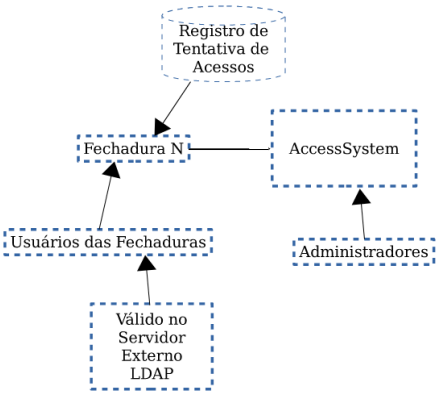

Figura 1. Esquema abstrato

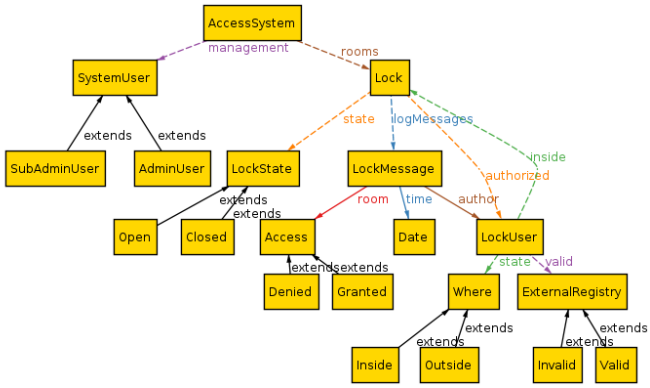

Figura 2. *Metamodelo*

Na **Figura 1** é mostrado o modelo abstrato, este modelo é uma concepção do sistema, a partir dele podemos afirmar que é composto por um conjunto com um número arbitrário de fechaduras, ele é administrado por administradores que podem incluir, listar ou remover elementos das fechaduras. Cada uma possui um conjunto de usuários associados, um determinado usuário só consegue ativar a fechadura se estiver na lista associada à mesma. Todas as tentativas de ativação, mesmo que sem sucesso, serão armazenadas em um banco de dados para posterior consulta. Todos os usuários da fechadura possuem uma propriedade que diz se seu registro no sistema está ativo ou não. Esta propriedade deve ser consultada toda vez que houver uma tentativa de ativação de uma fechadura ou quando o usuário for adicionado como permitido o acesso em uma sala. Existe apenas uma situação em que o usuário está associado à sala mas não possui registro válido, acontece quando a permissão já existe mas seu registro foi excluído do banco de registros LDAP (ex.: em caso de desligamento da organização).

A formalização em *Alloy* está na **Figura 2**, onde é mostrado o *Metamodelo*, isto é, um diagrama de assinaturas (ou classes) e relacionamentos mostrando como os elementos estão diretamente interagindo. Este modelo possui alumas propriedades a serem garantidas, como por exemplo, apenas pode entrar em uma sala quem é autorizado e com registro válido, além de estar associado àquela sala. Se não estiver associado a uma sala, não é possível entrar.

Para trazer o sistema do modelo abstrato para o físico, foi utilizado uma arquitetura conforme mostrado na **Figura 3.**

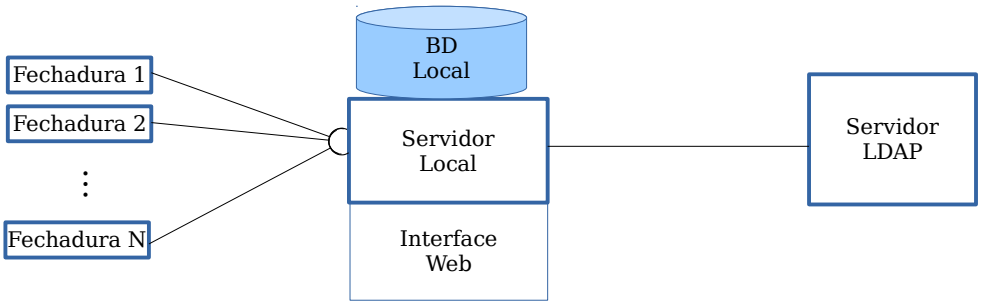

Figura 3. Arquitetura Geral do Sistema

O sistema que gerencia as permissões é composto pelo Servidor Local, Banco de Dados Local e a Interface web, usada para o gerenciamento. As fechaduras são os clientes deste sistema, mostradas à esquerda na **Figura 3**. O sistema também é um cliente do banco de dados LDAP, que atesta se um usuário tem seu registro válido ou não. Todas as comunicações entre as partes do sistema é feita através da rede. Abaixo as características de cada parte separadamente:

2.1. Fechaduras

As fechaduras são dispositivos com capacidade de enviar requisições via HTTP, inclusive usando o protocolo TLS para criptografia. Elas executam um sistema operacional de propósito geral baseado em Linux. Cada fechadura possui uma associação com a respectiva sala a qual foi configurada para receber os dados do identificador do usuário. Estes dados podem ser obtidos da carteirinha. O dado determinante que é associado a cada usuário da fechadura que possui a carteirinha é o registro (ou matrícula), este identificador é único perante os usuários cadastrados na organização. O cartão possui a credencial que informa de forma segura este dado para a Fechadura. A partir daí temos o par (sala, matrícula) e podemos verificar junto ao Banco de Dados Local se determinada matrícula está associada à determinada sala.

As fechaduras foram desenvolvidas pelo VerLab (*Laboratory of Computer Vision & Robotics*) da UFMG e pode ser encontrada em [2]. Esta fechadura utiliza equipamentos de baixo custo. Seus principais componentes são: um Raspberry Pi, um leitor RFID e um relé. O leitor RFID lê o cartão do usuário, o Raspberry Pi recebe os dados da leitura e realiza uma consulta ao servidor local para verificar a associação e validade do usuário. Em seguida, o servidor responde à solicitação, e em caso de concessão do acesso, o Raspberry Pi ativa o relé e a fechadura se abre para o usuário.

2.2. Servidor Local

O servidor local é uma implementação de uma API REST com um sistema para gerenciar as permissões de usuários utilizando *Python* na versão 3.6. Abaixo temos a lista dos pacotes mais importantes que foram utilizados neste sistema:

- *Flask*: *Framework* minimalista para criação de sistemas web com gerador de templates em HTML[3];
- *SQL-Alchemy*: Interface para declaração de banco de dados usando paradigma de orientação a objetos;
- *Gunicorn* – Pacote em python que é um servidor *WSGI (Web Server Gateway Interface)*. Este tipo de servidor é usado para adicionar paralelismo no atendimento aos clientes, além de servir conteúdo estático de forma mais rápida. Este servidor em específico adiciona paralelismo de processos para tratar as requisições.
- LDAP – Foram utilizados alguns pacotes para interação com o LDAP, o *python-ldap* é uma biblioteca em usada para comunicação com serviços LDAP. Foi utilizado também o ldap3 que possui a funcionalidade de estabelecer um servidor LDAP, que neste caso foi utilizado para testes.

- *Docker* - O servidor *WSGI* será embutido em uma imagem *Docker*. Este passo é feito para facilitar a configuração do serviço no ambiente de produção.

O Servidor Local possui duas interfaces de acesso: a API REST e a Interface Web.

A API rest é a interface para que as fechaduras façam requisições. Ela possui apenas um recurso que pode ser acessado no caminho ***/api*** e deve receber requisições HTTP POST. A **Tabela 1** descreve o formato do dado a ser enviado e recebido por um cliente ao interagir com o Sistema. A API deve receber os dados codificados na sintaxe JSON com o cabeçalho "*Content-Type: application/json*" e deve esperar um par como resposta.

	Nome do Campo	Tipo do Valor
Requisição feita pela fechadura	*room*	*String*
	user	*User Object*
	~~***auth***~~	~~***String***~~
Resposta do servidor local	*result*	*Integer*
	~~***auth***~~	~~***String***~~
	message	*String*

Tabela 1. Requisições à API

O campo *room* é uma uma sequência de caracteres que foi cadastrada previamente no servidor local, ela identifica unicamente uma sala, neste caso será o número da sala.
O campo *user* deve receber um objeto que represente um usuário da fechadura. Anteriormente foi assumido que o usuário possui mais dados além da matrícula, como dados de autenticação, mas para simplificar podemos considerar apenas a matrícula a ser transmitida neste campo.
O campo *auth* é uma sequência de 128 caracteres em hexadecimal usada na autenticação de uma mensagem tanto na requisição quanto na resposta. Ele foi removido neste trabalho para questões de simplificação e porque não é necessário. Uma vez que o servidor fará o uso de certificados TLS, a comunicação se torna criptografada e o forjamento de servidores se torna impossível, desde que a chave privada do certificado do servidor não esteja comprometida. Não existe a garantia de que o cliente não possa ser forjado, pois o TLS exige que apenas o servidor seja validado por padrão. Mas é possível obrigar o cliente a apresentar um certificado válido, através de uma simples configuração no servidor. Desta forma o servidor faz o desafio para o cliente para verificar o certificado e não prossegue com as requisições caso o cliente seja inválido.
O campo *result* é um valor representando um inteiro que pode ser *0*, indicando para não abrir a porta; e *1*, indicando para abrir a porta.
O campo *message* é uma sequencia que informa qual é o motivo da fechadura não ser ativada pelo usuário, existem duas possibilidades para este caso: O usuário não está registrado na organização ou o usuário está registrado mas não tem a permissão para acessar a sala.

O banco de dados do Servidor Local armazena 4 tipos de usuários, cada um deles está listado abaixo, junto com suas características e tipo de identificador:

- Administrador: sempre existirá este usuário no sistema, apenas a ele é permitido criar ou remover fechaduras e sub administradores além de conceder ou revogar permissões. O identificador *'0'* é reservado para este usuário;
- Sub administradores: são administradores que podem ser criados de forma arbitrária pelo Administrador. Eles tem poder apenas para conceder e revogar permissões. Pode ser atribuído qualquer identificador para estes usuários, desde que não seja *'0'* ou esteja na faixa de registro dos usuários das fechaduras, que possuem um identificador em formato de string com 10 caracteres, nem das fechaduras, que possuem 4 dígitos;
- Fechaduras: são adicionadas pelo administrador e representam a sala, seus identificadores devem possuir 4 dígitos;
- Usuários das fechaduras: eles possuem o identificador composto por uma sequencia de 10 caracteres numéricos. Eles são adicionados ao Servidor Local toda vez que um usuário recebe uma permissão de acesso, e são removidos quando não há permissão de acesso do usuário para alguma sala.

A interface web possui login o qual apenas os administradores podem entrar. Ela pode ser acessada pelo navegador e possui as funcionalidades para os administradores gerenciarem usuários e permissões. Além disso existe uma página de ajuda informando como usar o sistema. Ela foi construída usando as ferramentas de *templates* do *framework Flask*, o código HTML é gerado de forma automática e dinâmica. Não existe nenhum código em *JavaScript*. Esta decisão foi tomada para tornar o sistema leve e compatível com navegadores que não executam *JavaScript*. Diferente de uma abordagem que abusa de *frameworks* complexos e adicionam muitos elementos consumindo muitos recursos da máquina.

Toda tentativa de acesso às salas por usuários, devem ser registrados no Servidor Local, serão gravadas a data, hora, sala e se foi permitida a entrada ou não. Isto será feito para fins de controle de acesso, para posterior consulta em caso de alguma ocorrência na sala. Os administradores podem consultar os registros desses acessos por meio do Servidor Local, que podem ser filtradas por sala.

2.3. Servidor LDAP

O servidor LDAP é um banco de dados da organização para armazenamento e consulta de objetos que são utilizados para representar pessoas e inclusive coisas, como uma sala, um programa de computador, um domínio na internet, etc. Este banco de dados se estrutura em forma de árvore e os usuários registrados nele possuem um identificador único. A consulta do Servidor Local no LDAP se dá mediante ocorrência de dois eventos: toda vez que um usuário for associado à uma fechadura no Servidor Local, e toda vez que um usuário tentar ativar uma fechadura para abrir uma porta. A sua matrícula deve ser consultada no servidor LDAP, para fins de consistência. Inicialmente

pensou-se em criar objetos e escrever diretamente no servidor LDAP, porém como trata-se de um sistema crítico, porque deve manter o cadastro dos usuários de forma segura e consistente, muitas vezes as organizações impõe políticas para que esta forma de utilização fique limitado ao departamento responsável. Esta questão pode ser resolvida através da criação das associações no Servidor Local e da utilização do servidor LDAP externo apenas para consulta.

3. Implementação

O código pode ser encontrado em [4]. Ele contém um *Makefile* que ao digitar ***make***, é exibida uma pequena ajuda. O sistema foi desenvolvido utilizando o Github para controle de versões e possui testes de unidade para garantir algumas propriedades do banco de dados, além de um teste de sistema para atestar o seu funcionamento.

Foi construída uma imagem *Docker* que está disponível em [5]. Ela é construída automaticamente usando o *Github Actions*, após enviar um *commit* para o *branch docker* do repositório.

4. Trabalhos futuros

Como trabalhos futuros, eu pretendo estender este sistema para outras funcionalidades e casos de uso. Uma possível funcionalidade seria acrescentar possibilidade de verificação biométrica, usando imagens, impressões digitais ou até mesmo a voz.

Um caso de uso interessante, seria estender este sistema para acrescentar funcionalidades de um interfone com vídeo porteiro para prédio de apartamentos, neste caso teria um teclado para a possibilidade de comunicação com um número arbitrário de apartamentos. Os usuários seriam cadastrados no banco de dados local, ou até poderiam usar um serviço de terceiro. E o acesso à um apartamento seria feito da mesma forma descrita neste trabalho.

5. Referências

[1] *Alloy Analyzer* https://alloytools.org/about.html

[2] Repositório *VerLab* https://github.com/h3ct0r/VerlabAccessSystemRPi

[3] *Framework Flask* https://flask.palletsprojects.com/en/2.3.x/

[4] Repositório *access-system* https://github.com/lcharles123/access-system

[5] Imagem *Docker* https://hub.docker.com/repository/docker/lcharles060/access-system/general