Instituto Federal de Educação Ciência e Tecnologia do Maranhão
Departamento de Computação – DCOMP
Disciplina: Laboratório de Desenvolvimento de Software
Professor: Mauro Lopes C. Silva

Sistema Chavoso

Documento de Requisitos

Equipe:
201612030300 – David Alysson Cotrim Marques
201612030327 – Elias Rocha Lima
201612030289 – Samuel de Oliveira
201612030564 – Alexsandro Soares Teixeira

São Luís – MA

Sumário

1. Organização do Documento

O documento está organizado em descrição do sistema, requisitos funcionais, requisitos não funcionais, restrições, usuários e diagramas.

2. Descrição do Sistema

Este Sistema é destinado a controlar a entrega e o recebimento das chaves dos laboratórios dos departamentos de ensino técnico do Instituto Federal de Educação, Ciência e Tecnologia campus Monte Castelo. O sistema deverá ser utilizado pela recepção e dessa forma os recepcionistas serão os administradores do software, devendo então se responsabilizar pela entrega e devolução das chaves dos laboratórios.

No momento em que uma pessoa desejar pegar uma chave, será necessário uma assinatura digital que confirme a identidade da pessoa, caso a pessoa tenha permissão para receber a chave está será entregue. Para devolver uma chave não é feito a verificação, é apenas salvo o horário e a data da devolução.

Os professores têm total liberdade de pegar qualquer chave, desde que o laboratório esteja em seu departamento e disponível, os auxiliares de limpeza tem permissão para pegar a chave de qualquer laboratório disponível.

Um laboratório pode ter quatro estados distintos, sendo eles: “livre”, “ocupado’’, “em limpeza” e “em manutenção”. Um laboratório é dito como ‘’ocupado’’ quando estiver sendo usado por algum professor, é dito como ‘’em limpeza’’ quando estiver sendo realizado uma higienização do mesmo pela equipe de limpeza do campus, será considerado ‘’em manutenção’’ aquele laboratório que esteja impossibilitado de ser utilizado devido a problemas , em sua estrutura, que necessitem de uma reforma ou enquanto estiver sendo reformado. Um laboratório ‘’livre’’ é aquele que está disponível, e capacitado, para ser utilizado por professores ou auxiliares de limpeza naquele momento.

3. Requisitos

3.1 Requisitos Funcionais

RF001 –Manutenção de professor

O sistema deverá ser responsável pela manutenção de professores, de forma que deverá ser capaz de cadastrar, atualizar e armazenar informações responsáveis pela identificação do professor.

Prioridade: (**x**) Essencial ( ) Desejável

RF002 –Manutenção de auxiliares de limpeza

O sistema deverá ser responsável pela manutenção de auxiliares de limpeza, sendo assim responsável por permitir o cadastro, a atualização de informações e o armazenamento dessas informações para que seja possível identificar o auxiliar.

Prioridade: (**x**) Essencial ( ) Desejável

RF003 – Manutenção de laboratório

O sistema deverá ser responsável pela manutenção de laboratórios, de formar que deverá cadastrar as informações de novos laboratórios, atualizar e garantir o armazenamento de dados de laboratórios já cadastrados.

Prioridade: (**x**) Essencial ( ) Desejável

RF004 – Manutenção de departamento

O sistema deverá ser responsável pela manutenção de departamentos, de forma que deverá permitir o cadastro e armazenamento de novos departamentos assim como a atualização de suas informações.

Prioridade: (**x**) Essencial ( ) Desejável

RF005 - Manutenção de administrador

O sistema deverá ser responsável pela manutenção do administrador, de forma que será capaz de cadastrar novos administradores, permitir a atualização dos dados desses administradores e ,principalmente, garantir o armazenamento desses dados.

RF006 - Manutenção de histórico

O sistema deverá ser responsável por inserir e armazenar as ações realizadas no sistema. De forma que, caso seja necessário, possam ser identificados os responsáveis por acontecimentos no sistema.

3.2 Requisitos Não Funcionais

RNF001 – Requisitos de manutenção

Categoria: Integridade

Descrição: O sistema utilizará professores, laboratórios e departamentos já pré-definidos, baseando-se na estrutura e quadro de funcionários atual da instituição.

Prioridade: (**x**) Essencial ( ) Desejável

RNF002 – Requisitos de acesso

Categoria: Segurança

Descrição: O sistema utilizará login e senha para garantir o acesso de um administrador e uma assinatura digital para garantir a integridade da entrega de uma chave para um professor ou auxiliar de limpeza.

Prioridade: (**x**) Essencial ( ) Desejável

RNF003 – Requisitos de segurança

Categoria: Segurança

Descrição: Apenas o administrador pode entregar ou receber as chaves do laboratório, desde que o professor confirme a sua identidade.

Prioridade: (**x**) Essencial ( ) Desejável

3.3 Restrições

RE001 – Retenção de um laboratório

Descrição: Baseado no sistema atual de disponibilidade dos laboratórios, um laboratório só pode ser disponibilizado caso a chave esteja sobre posse do administrador. Logo o sistema deverá impedir o acesso a laboratórios já ocupados e/ou indisponíveis por motivos diversos(Limpezas,reformas,etc).

Dependência: RF005

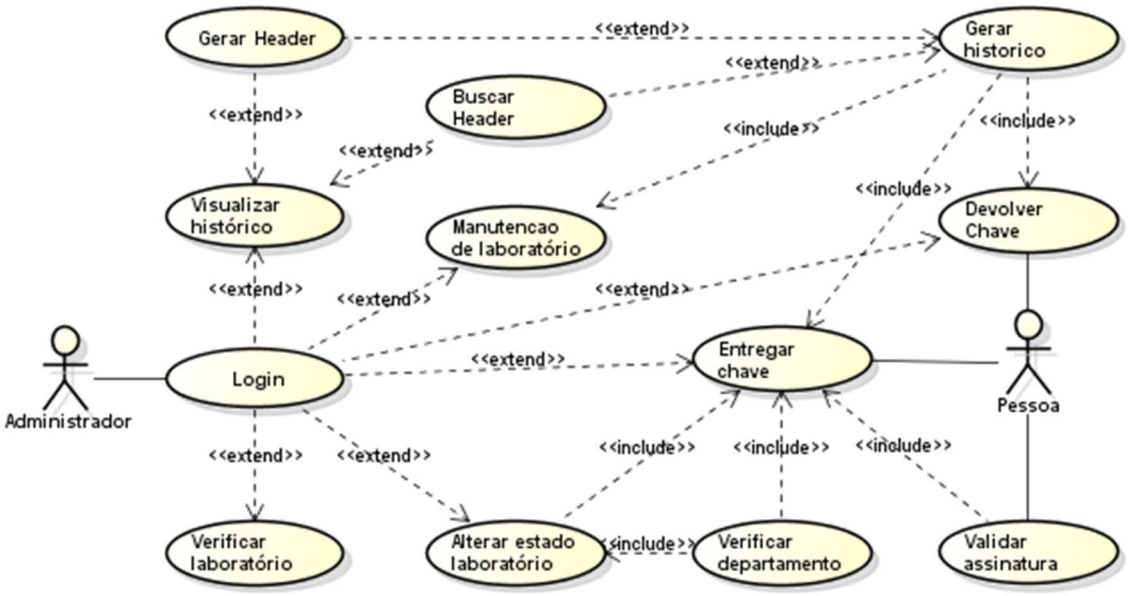
4. Usuários

Recepcionista: Administrador do sistema responsável por entregar, receber e armazenar as chaves dos laboratórios, indicar períodos de reforma e o único com autorização para efetuar login no sistema.

Professor: Usuário do sistema, será o responsável por pegar uma chave para utilizar o laboratório e em seguida liberar o laboratório realizando a devolução da chave.

Auxiliar de limpeza: Responsável pela manutenção dos laboratórios, pode pegar a chave e ocupar os laboratórios para realizar a limpeza de tal. Após a limpeza devolverá a chave e o sistema irá disponibilizar novamente o laboratório.

5. Diagrama de caso de uso



- **Caso de Uso: Login ()**

Descrição: Se os dados preenchidos no login estiverem cadastrados no sistema e estiverem certos, este caso de uso possibilitará o acesso ao programa;

Requisitos e Regras de Negócio: RF005 e RNF002

Classes: Login, Administrador

Ator Primário: Administrador

Pré-Condição: O Administrador deve estar cadastrado no sistema;

Pós-Condição: O Administrador terá acesso ao sistema e suas funções;

Fluxo Principal:

1. O sistema auto-inicia a tela de Login();
2. A tela requisita o CPF do administrador e a sua SENHA, dados necessários para o login;
3. O Administrador informa os seus dados nos campos indicados;
4. O Sistema gera uma conexão com o banco de dados para iniciar a verificação dos dados;
5. O Sistema verifica se o CPF está cadastrado no sistema;
6. O Sistema verifica se a SENHA informada é igual àquela associada ao Administrador dono do CPF;
7. O Administrador consegue o acesso ao sistema;
8. É iniciada a tela principal do Sistema e fechada a tela de Login();

Fluxo Alternativo:

1. Para os Fluxos Principais 1 e 2:

    1. É iniciada na Tela uma mensagem, em uma janela do JOptionPane, informando qual foi o erro e se possível o que causou esse erro ao iniciar a tela e/ou seus componentes.

    2. Todo o Sistema terá sua execução finalizada.

2. Para o Fluxo Principal 3:

    1. O Sistema deverá manter a tela pronta para receber os dados, independente do tempo que um administrador leve para informá-los.

3. Para o Fluxo Principal 4:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela o erro gerado ao tentar acessar o banco

    3. A tela de Login() deverá ser reiniciada

4. Para o Fluxo Principal 5:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que o CPF digitado não foi localizado no banco

    3. A tela de Login() deverá ser reiniciada

5. Para o Fluxo Principal 6:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que a SENHA digitada é igual àquela associada ao CPF

    3. O Sistema irá encerrar a conexão com o banco de dados.

    4. O campo de SENHA da tela de Login() deverá ser limpo

5. O sistema irá requisitar que seja digitado novamente a senha(Retornando assim ao Fluxo Principal 3)

6. Para o Fluxo Principal 7:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela o motivo pelo qual não foi permitido o acesso.

    3. A tela de Login() deverá ser reiniciada

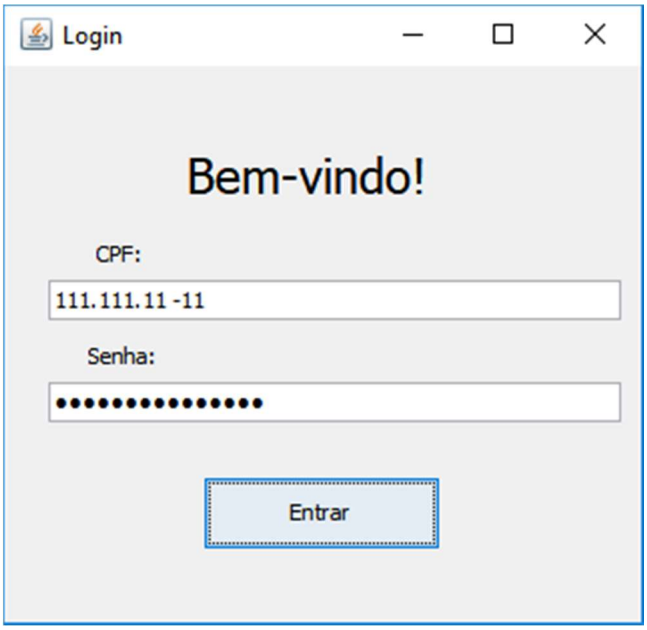
7. Para o Fluxo Principal 8:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não foi possível inicializar a tela inicial e se possível, por meio de Exceptions, informar o erro gerado

    3. A tela de Login() deverá ser reiniciada

Protótipo da Tela associado a este Caso de Uso:



- **Caso de Uso: VerificarDisponibilidadeLaboratório()**

Descrição: Esse caso de uso serve para verificar o estado de algum laboratório, ou seja, analisar se o laboratório está disponível ou não.

Requisitos e Regras de Negócio: RF004, RNF001, RE001

Classes: Administrador, Laboratório, Gerenciar_historico, Relatório

Ator Primário: Administrador;

Pré-Condição: Login() deve ter sido feito e o laboratório escolhido pelo administrador deverá ser válido;

Pós-Condição: O laboratório deverá ter seu estado analisado pelo sistema;

Fluxo Principal:

1. O administrador abre o menu “Laboratórios” da tela Principal do Sistema e clica no item de menu “Verificar a disponibilidade”;
2. O Sistema irá abrir uma Janela ou apenas sobrepor caso exista uma Janela igual a essa já aberta, utilizando a classe JInternalFrame do Javax.swing, com os campos necessários para que o administrador informe o laboratório que deseja verificar
3. O Sistema irá abrir uma conexão com o banco de dados para ter acesso aos dados dos laboratórios
4. O administrador irá escolher primeiramente um departamento em uma lista de itens de um JComboBox e em sequência poderá escolher um laboratório do departamento escolhido, novamente a partir de um JComboBox.
5. O Sistema irá verificar os dados desse laboratório.
6. O sistema irá verificar se já existe alguma janela informando a disponibilidade de um laboratório e irá optar entre fechá-la ou cancelar a verificação atual(Irá aparecer uma JOptionPane para que ele confirme o fechamento da tela ou a interrupção da verificação, não é permitido que várias janelas desse gênero estejam abertas simultaneamente).
7. O sistema irá iniciar outra Janela (JInternalFrame), essa deverá ser sobreposta à janela criada ao clicar no item de menu “verificar a disponibilidade” do menu “Laboratórios” e ela irá informar o número do laboratório, seu departamento, a localização da sua chave e a sua disponibilidade atual.

Fluxo Alternativo:

1. Para o Fluxo Principal 2:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não foi possível inicializar a janela de verificação e se possível, por meio de Exceptions, informar o erro gerado

    3. A tela principal deverá ser reiniciada

2. Para o Fluxo Principal 3:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não foi possível estabelecer conexão com o banco de dados e se possível, por meio de Exceptions, informar o erro gerado

    3. A janela de verificação de disponibilidade do laboratório deverá ser reiniciada

3. Para o Fluxo Principal 4:

    1. O Sistema deverá se manter pronto para receber os dados, independentemente do período de tempo que o administrador levar para informá-los.

    2. Caso tenha algumas inconsistências nos dados informados, o Sistema irá informar em uma janela do JOptionPane que é necessário que o Administrador informe novamente os dados e se possível informe também o que levou a essa inconsistência. Após o usuário confirmar na caixa de diálogo o fluxo principal retornará para o Fluxo Principal 3.

4. Para o Fluxo Principal 5:

1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não foi possível analisar os dados retornados pelo banco de dados e se possível, por meio de Exceptions, informar o erro gerado

3. A janela de verificação de disponibilidade do laboratório deverá ser reiniciada

5. Para o Fluxo Principal 6:

1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não foi possível iniciar a Janela com a disponibilidade do laboratório e se possível, por meio de Exceptions, informar o erro gerado

3. A janela de verificação de disponibilidade do laboratório deverá ser reiniciada

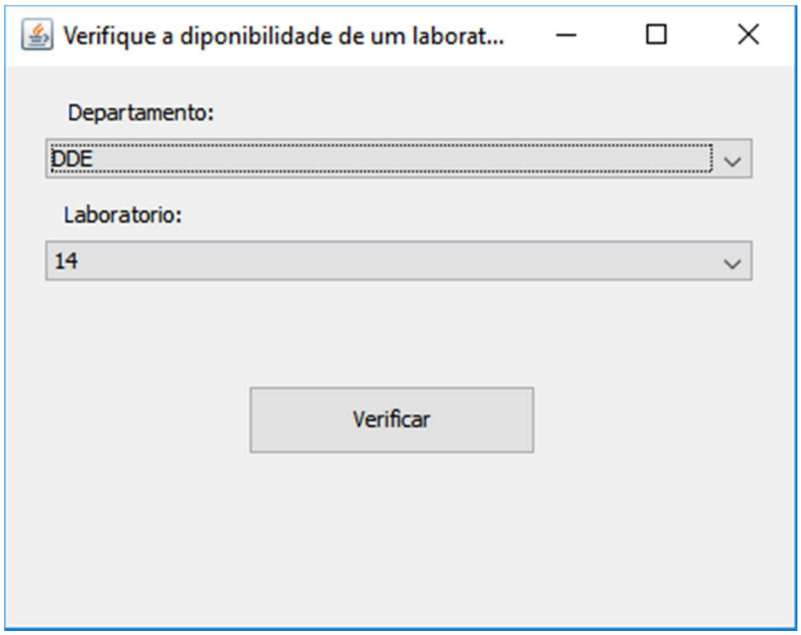
6. Para o Fluxo Principal 7:

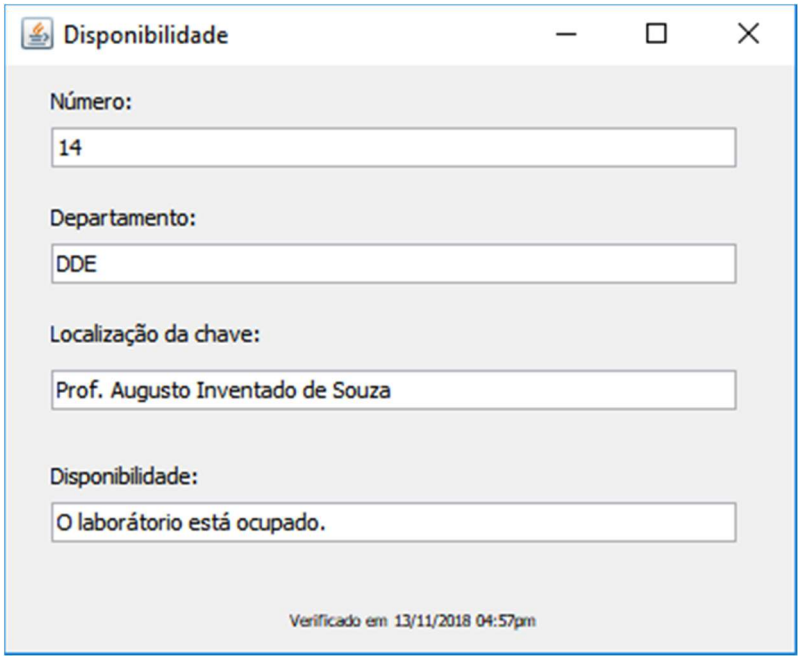
1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não foi possível iniciar a nova janela e se possível, por meio de Exceptions, informar o erro gerado

3. A janela de verificação de disponibilidade do laboratório deverá ser reiniciada

Protótipos da Tela associado a este Caso de Uso:





- **Caso de Uso: Alterar disponibilidade Laboratorio ()**

Descrição: Esse caso de uso altera a disponibilidade do laboratório entre "disponível", "ocupado", "Em manutenção" ou "Em limpeza".

Requisitos e Regras de Negócio: RF003, RE001

Classes: Laboratório

Casos de Uso: Entregar chave(), Devolver chave(), Manutencao()

Ator Primário: Administrador, Professor, Auxiliar de limpeza;

Pré-Condição: Login() deve ter sido feito e caso essa alteração tenha sido solicitada pelo caso de uso "Entregar Chave()" a pessoa(Professor ou Auxiliar) devem ter assinado confirmando a alteração.

Pós-Condição: O estado do laboratório deverá ser alterado;

Fluxo Principal (1):

1. O caso de uso será chamado pelo caso de uso "Manutenção ()"
2. O caso de uso irá receber um laboratório e uma disponibilidade
3. O sistema deverá gerar uma conexão com o banco de dados para o caso de uso
4. O caso deverá enviar um pacote SQL para o banco alterando a disponibilidade do laboratório para a disponibilidade desejada
5. O caso de uso deverá retornar a confirmação de que a disponibilidade foi atualizada

Fluxo Alternativo (1):

1. Para os fluxos principais 1,2,3,4: Uma mensagem via JOptionPane deverá informar o erro e retornar como "falha ao atualizar" para o caso de uso "Manutenção"

2. Para o fluxo principal 5: Uma mensagem via JOptionPane deverá informar que a disponibilidade foi alterada e, ao mesmo tempo, informar que o sistema deve ser reiniciado, em sequência o sistema deverá reiniciar fechando todas as janelas abertas e abrindo uma nova janela “Principal”

Fluxo Principal (2):

1. O Administrador, após confirmar na janela “Devolução de Chave” que alguém devolveu a chave o caso de uso “Devolver chave()” chama o caso de uso “Alterar disponibilidade laboratório” para que o respectivo laboratório selecionado na janela seja colocado como “disponível”
2. Já no atual caso de uso, o Sistema deverá criar uma conexão com o banco de dados
3. O Sistema irá enviar para o banco de dados um pacote SQL solicitando o update da coluna disponibilidade do respectivo laboratório para “disponível”
4. O Sistema deverá encerrar a conexão com o banco de dados
5. Será retornado para o caso de uso “Devolver chave()” a confirmação da atualização

Fluxo Alternativo (2):

1. Para o Fluxo Principal 2:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não foi possível comunicar-se com o banco e se possível, por meio de SQL Exceptions, informar o erro gerado

    3. O caso de uso deverá retornar para “Devolução de Chave” que o banco está indisponível.

2. Para o Fluxo Principal 3:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não foi possível atualizar a coluna “disponibilidade” da tabela “Laboratorios” na linha referente ao respectivo laboratório e se possível, por meio de SQL Exceptions, informar o erro gerado.

    3. O caso de uso deverá retornar para “Devolução de Chave” que o banco está indisponível.

3. Para o Fluxo Principal 4:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que existe um erro na comunicação com o banco e se possível, por meio de SQL Exceptions, informar o erro gerado

    3. O caso de uso deverá retornar para “Devolução de Chave” que atualizou as informações mas um erro ocorreu necessitando que sua alteração seja verificada e, caso tenha sido incorreta, corrigida.

4. Para o Fluxo Principal 5:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que atualizou a disponibilidade e se possível, por meio de Exceptions, informar o erro gerado ao tentar retornar a confirmação

    3. O caso de uso deverá reiniciar “Devolver Chave()” e retorna novamente a confirmação sem realizar a repetição do processamento.

Fluxo Principal (3):

1. Após ser confirmada a assinatura de um professor/auxiliar de limpeza na janela “Entrega de Chave”, o caso de uso “Entregar chave()” chama o caso de uso “Alterar disponibilidade laboratório” para que passando um parâmetro que determinará qual será a nova disponibilidade(“Ocupado” caso a assinatura seja de um professor ou “Em limpeza” caso a assinatura seja de auxiliar de limpeza).
2. Já no atual caso de uso, o Sistema deverá criar uma conexão com o banco de dados
3. O Sistema irá enviar para o banco de dados um pacote SQL solicitando o update da coluna disponibilidade do respectivo laboratório para, baseado no parâmetro passado, “Ocupado” ou “Em limpeza”
4. O Sistema deverá encerrar a conexão com o banco de dados
5. Será retornado para o caso de uso “Entregar chave()” a confirmação da atualização

Fluxo Alternativo (3):

1. Para o Fluxo Principal 2:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não foi possível comunicar-se com o banco e se possível, por meio de SQL Exceptions, informar o erro gerado

    3. O caso de uso deverá retornar para “Entrega de Chave” que o banco está indisponível.

2. Para o Fluxo Principal 3:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não foi possível atualizar a disponibilidade do laboratório no banco e se possível, por meio de SQL Exceptions, informar o erro gerado

    3. O caso de uso deverá retornar para “Entrega de Chave” que não houve a atualização desejada.

3. Para o Fluxo Principal 4:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que existe um erro na comunicação com o banco e se possível, por meio de SQL Exceptions, informar o erro gerado

3. O caso de uso deverá retornar para “Entrega de Chave” que atualizou as informações mas um erro ocorreu necessitando que sua alteração seja verificada e, caso tenha sido incorreta, corrigida.

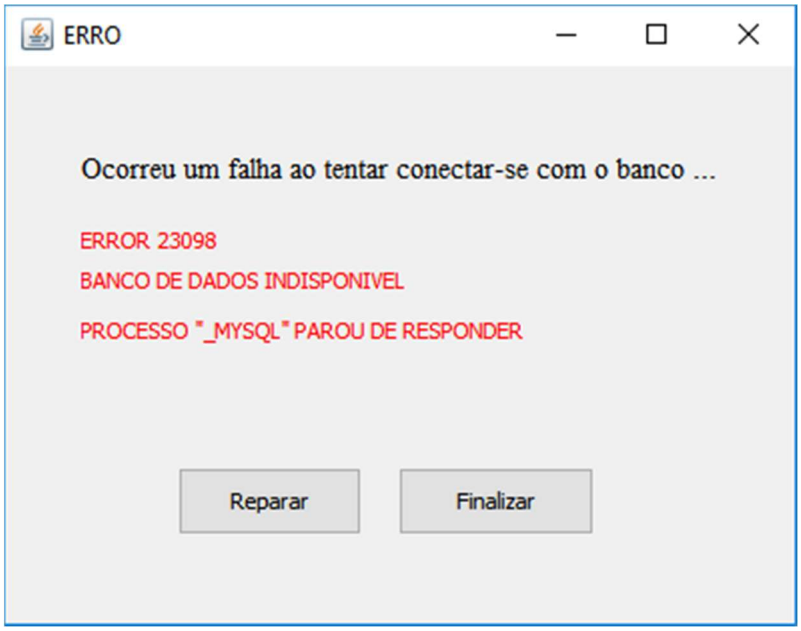
4. Para o Fluxo Principal 5:

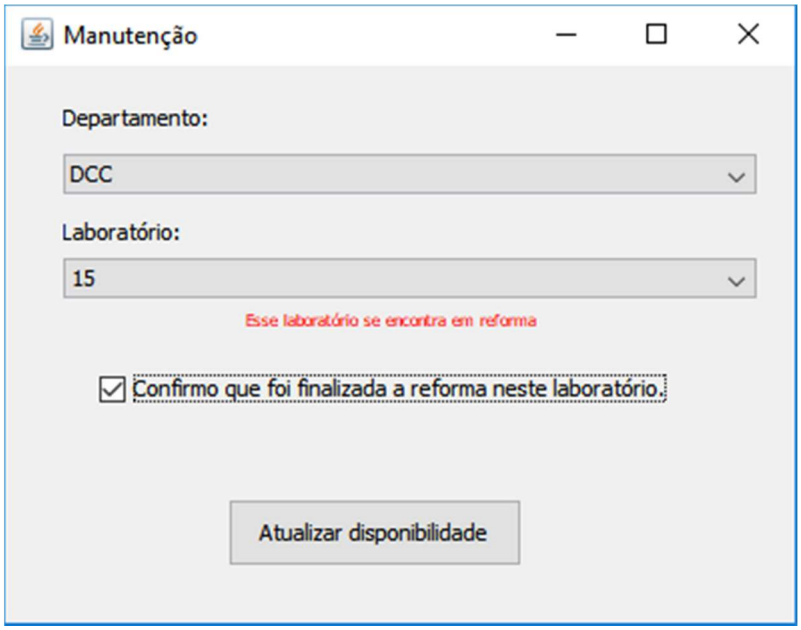
1. O Sistema iniciará uma janela do JOptionPane

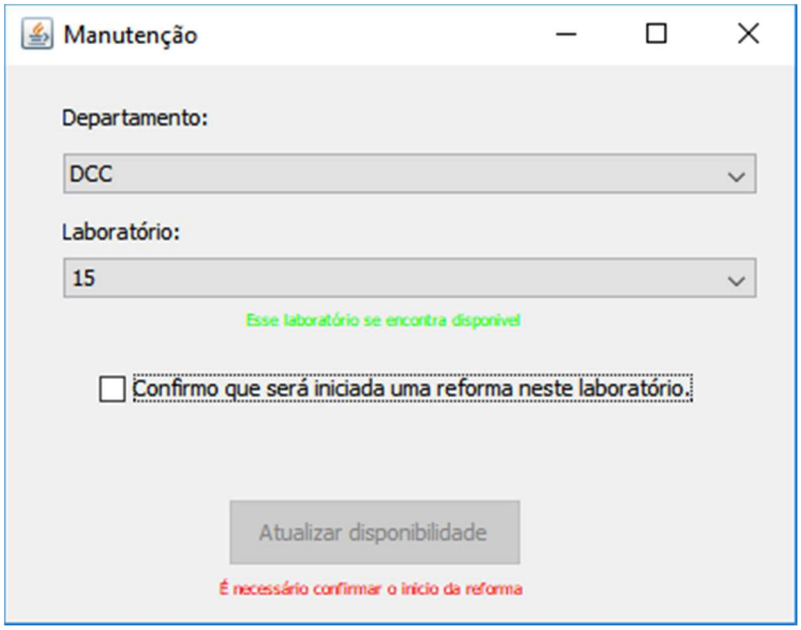
2. O Sistema irá informar na janela que atualizou a disponibilidade e se possível, por meio de Exceptions, informar o erro gerado ao tentar retornar a confirmação

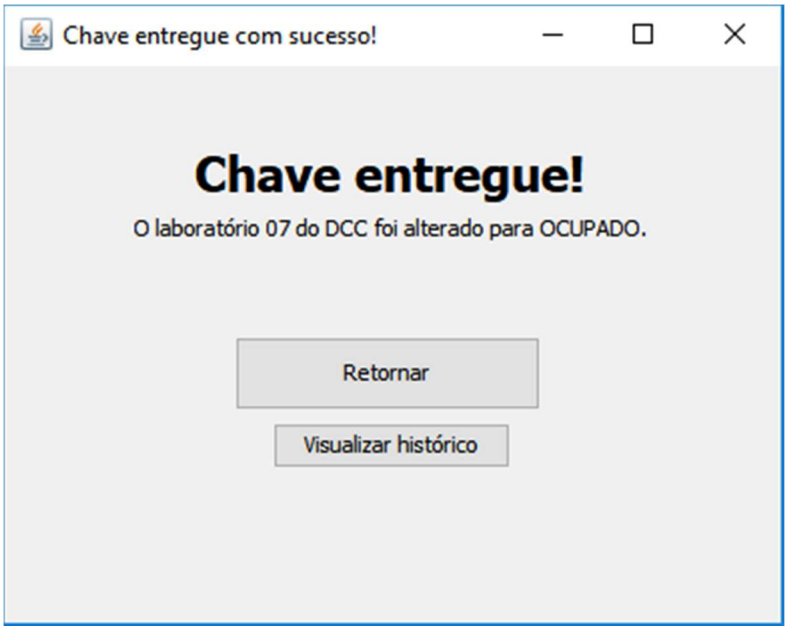
3. O caso de uso deverá reiniciar “Entrega de Chave()” e retorna novamente a confirmação sem realizar a repetição do processamento.

Protótipo(s) da Tela associado a este Caso de Uso:









*esse último protótipo é de “Entrega chave”), mas é nessa interface que aparecerá o retorno do fluxo principal (3).

**essa mesma situação pode ser vista no caso de “Devolver Chave()”, tendo apenas alterações na mensagem e composição do protótipo

- **Caso de Uso: Entregar Chave()**

Descrição: Ao entregar a chave, o estado do laboratorio irá mudar automaticamente para “ocupado” ou “Em limpeza”.

Requisitos e Regras de Negócio: RF001, RF002, RF003, RF004, RF005, RNF003, RE001

Classes: Chave, Laboratório, Relatório, Professor, Auxiliar de Limpeza, Administrador

Casos de uso: Alterar disponibilidade(), verificar departamento(), validar assinatura()

Ator Primário: administrador;

Ator Secundário: Pessoa;

Pré-Condição: Login deve ter sido feito, Laboratório desejado deve estar com a disponibilidade “disponível”

Pós-Condição: A pessoa irá ter acesso ao laboratório, recebendo assim a chave deste

Fluxo Principal:

1. O sistema deverá iniciar uma janela “Entrega de Chave”, utilizando a classe JInternalFrame do javax.swing, assim que o administrador clicar no item de menu “Entregar Chave” do menu “Chaves”
2. O sistema deverá iniciar uma conexão com o banco de dados;
3. O sistema deverá recolher dados dos Departamentos e Laboratórios
4. O sistema preencherá os JComboBox da janela aberta para que o administrador consiga escolher um Laboratório.
5. O administrador deverá interagir com a interface selecionando o laboratório desejado
6. O sistema irá auto-verificar se o laboratório escolhido está com a disponibilidade “disponível”
7. O sistema irá abrir uma JOptionPane informando que é necessário que a pessoa que requisitou a chave digite sua assinatura
8. Após o Administrador clicar em OK na janela JOptionPane aberta, o Sistema deverá abrir uma pequena JinternalFrame para receber a assinatura.
9. O professor ou auxiliar de limpeza deverá digitar a assinatura na janela e confirmar(Clicando ENTER)
10. O sistema usará a conexão com o banco de dados para chamar o caso de uso “validar assinatura()” e, além disso, recuperar a função da pessoa no sistema
11. Caso seja um professor o sistema irá verificar se a chave requisitada é de um laboratório do seu departamento, utilizando para isso o caso de uso “verificar departamento()”.
12. O sistema irá iniciar o caso de uso “alterar disponibilidade laboratório()” e de acordo com a função da pessoa(Professor ou Auxiliar de limpeza) no sistema irá pedir para colocar como “ocupado” ou “em limpeza”
13. O sistema deverá finalizar a conexão com o banco de dados.
14. Assim que receber a confirmação que a disponibilidade for alterada o sistema irá informar por meio de uma JOptionPane para o Administrador que a chave pode ser entregue.

Fluxo Alternativo:

1. Para o fluxo Principal 1:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não foi possível iniciar a janela “Entrega de Chave” e se possível, por meio de Exceptions, informar o erro gerado, além disso, nessa janela terão dois botões: “reparar” e “finalizar”, respectivamente;

    3. O caso de uso deverá reiniciar “Entrega de Chave()” caso o Administrador clique em “reparar” ou deverá finalizar o fluxo principal de “entregar chave()” e reiniciar a janela “Principal” caso o Administrador clique em “finalizar” ou feche a janela(Clicando no “x” na parte superior da janela);

2. Para o fluxo Principal 2:

    1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não conseguiu se comunicar com o banco de dados e se possível, por meio de SQL Exceptions, informar o erro gerado

3. O fluxo deverá ser interrompido e finalizado.

3. Para o fluxo Principal 3:

1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não conseguiu recuperar as informações necessárias do banco de dados e se possível, por meio de SQL Exceptions, informar o erro gerado

3. O fluxo deverá ser interrompido e finalizado.

4. Para o fluxo Principal 4:

1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não conseguiu preencher a interface para utilização do administrador e se possível, por meio de Exceptions, informar o erro gerado

3. O fluxo deverá ser interrompido e finalizado.

5. Para o fluxo Principal 5:

1. O Sistema deverá se manter pronto para receber as informações

2. Caso o Administrador leve mais de cinco minutos sem interagir com a interface, o sistema deverá gerar uma janela JOptionPane que informará que o sistema está inativo e pedirá para que o administrador confirme.

3. O sistema deverá finalizar a conexão com o banco de dados

4. Apenas após o administrador confirmar a janela o fluxo deverá retornar para o fluxo principal 2.

6. Para o fluxo Principal 6:

1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não conseguiu verificar a disponibilidade e se possível, por meio de Exceptions, informar o erro gerado ou caso seja verificado que a disponibilidade seja diferente de “disponível” informar que o laboratório não está disponível.

3. O fluxo deverá ser interrompido e retornar para o fluxo 5.

7. Para o fluxo Principal 7:

1. O sistema deverá iniciar uma janela JInternalFrame informando a necessidade da assinatura e um botão para confirmar que o Administrador está ciente dessa assinatura.

2. Somente após o administrador clicar no botão confirmando está ciente, o sistema deverá gerar a JInternalFrame para receber a assinatura

3. O fluxo principal deverá pular para o fluxo 9

8. Para o fluxo Principal 8:

1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não conseguiu abrir a janela e se possível, por meio de Exceptions, informar o erro gerado

3. O fluxo deverá ser interrompido e iniciado o fluxo alternativo 7.

9. Para o fluxo Principal 9:

1. Caso a pessoa leve mais de 5 minutos para inserir a senha, o sistema deverá finalizar o fluxo e abrir uma janela JOPtionPane informando o erro “Tempo excedido”.

10. Para o fluxo Principal 10:

1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não conseguiu se comunicar com o banco de dados ou que não foi possível validar a senha e a função da pessoa. Se possível, por meio de SQL Exceptions, informar o erro gerado.

3. O fluxo deverá ser interrompido e finalizado.

11. Para o fluxo Principal 11:

1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que a pessoa não tem autorização para utilizar o laboratório daquele departamento

3. O fluxo deverá ser interrompido e finalizado.

12. Para o fluxo Principal 12:

1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não conseguiu alterar a disponibilidade do laboratório e se possível, por meio de Exceptions, informar o erro gerado

3. O fluxo deverá ser interrompido e finalizado.

13. Para o fluxo Principal 13:

1. O fluxo deverá ser pulado para o fluxo alternativo 14.

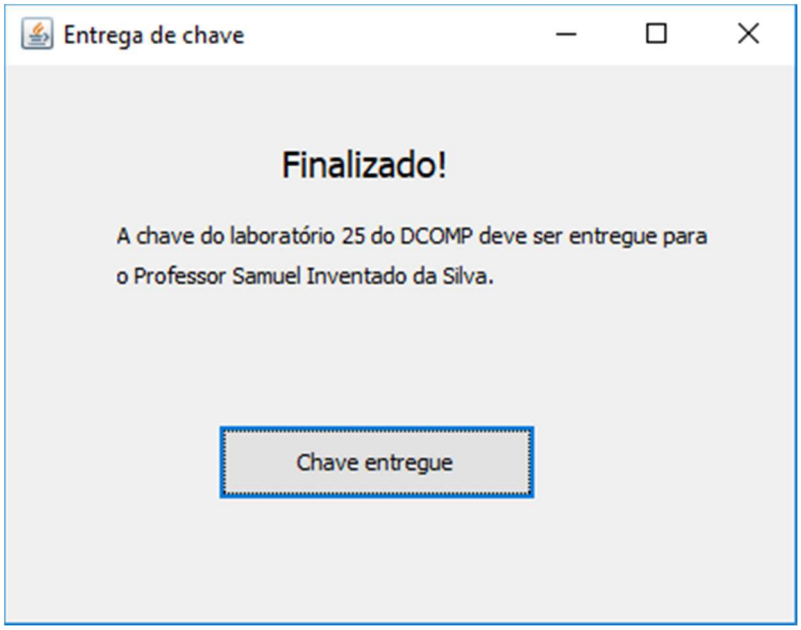
14. Para o fluxo Principal 14:

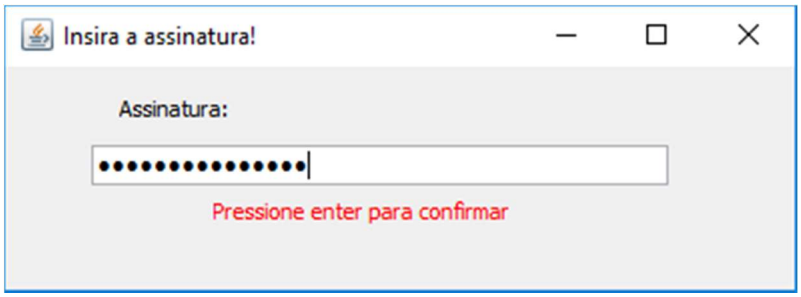
1. O Sistema iniciará uma janela JinternalFrame

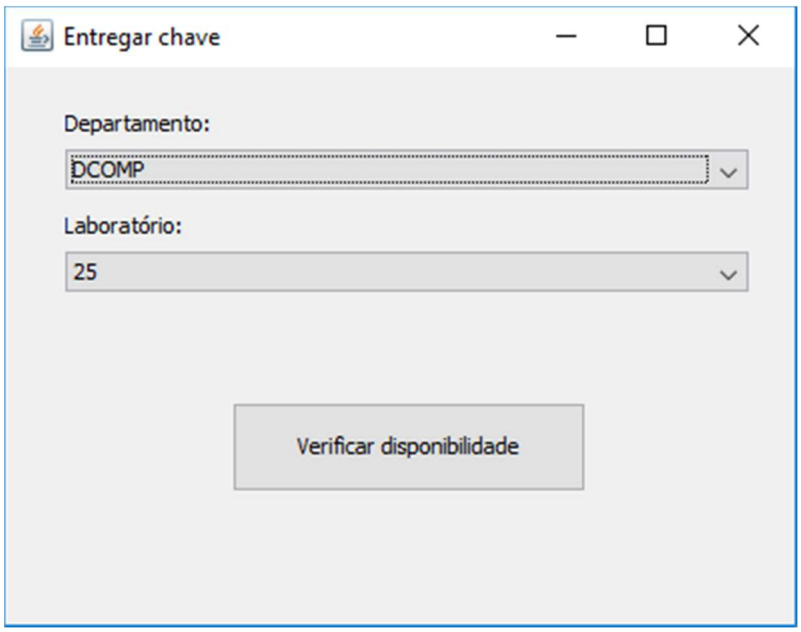
2. O Sistema irá informar na janela que o administrador deve entregar a chave à pessoa

3. O fluxo deverá ser interrompido e finalizado.

Protótipos da Tela associado a este Caso de Uso:







- **Caso de Uso: Devolver chave();**

Descrição: Ao devolver a chave, o estado do laboratorio irá mudar automaticamente para “Disponível”.

Requisitos e Regras de Negócio: RF001, RF002, RF003, RF005, RF006, RNF003

Classes: Administrador, Chave, Laboratório

Ator Primário: administrador;

Pré-Condição: Login() deve ter sido feito, deve existir pelo menos 1 chave sem estar sob posse do administrador

Pós-Condição: O estado do laboratório deve ser alterado e a chave ficará com o Administrador ;

Fluxo Principal:

1. O Administrador clica no item de menu “devolver chave” do menu “Chaves”;
2. O sistema deverá iniciar uma janela “devolver chave”, utilizando JInternalFrame do javax.swing
3. O sistema deverá iniciar uma conexão com o banco de dados
4. O sistema deverá buscar no banco as chaves que estão na condições de que podem ser devolvidas(aquelas de laboratórios com disponibilidade “ocupado” ou “em limpeza”)
5. O sistema deverá preencher os JComboBox da janela “devolver chave” com as chaves coletadas
6. A pessoa deverá entregar a chave para o Administrador
7. O Administrador deverá escolher qual chave está sendo devolvida(Escolhendo o laboratório)
8. O sistema deverá iniciar o caso de uso “alterar disponibilidade laboratório()” e pedir para tornar o laboratório “disponível”
9. O sistema deverá inicial uma janela JOptionPane e informar que o procedimento foi finalizado com sucesso.
10. A conexão com o banco de dados deverá ser finalizada.

Fluxo Alternativo:

1. Para o fluxo Principal 2:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não conseguiu iniciar a janela e se possível, por meio de Exceptions, informar o erro gerado.

    3. O fluxo deverá ser interrompido e finalizado.

2. Para o fluxo Principal 3:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não conseguiu criar uma conexão com o banco de dados e se possível, por meio de SQL Exceptions, informar o erro gerado.

    3. O fluxo deverá ser interrompido e finalizado.

3. Para o fluxo Principal 4:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não encontrou chaves para serem devolvidas e se possível , por meio de SQL Exceptions, informar o erro gerado.

    3. O fluxo deverá ser interrompido e finalizado.

4. Para o fluxo Principal 5:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não conseguiu preencher os campos da janela e se possível, por meio de Exceptions, informar o erro gerado.

    3. O fluxo deverá ser interrompido e finalizado.

5. Para o fluxo Principal 6:

    1. O fluxo deverá ser interrompido e finalizado.

6. Para o fluxo Principal 7:

    1. O Sistema deverá manter-se pronto para receber os dados informados pelo Administrador, independentemente do tempo que isso leve.

    2. Caso o sistema feche nessa etapa, ele deverá retornar para esse ponto na próxima inicialização.

7. Para o fluxo Principal 8:

    1. O Sistema deverá usar a conexão gerada para atualizar manualmente a disponibilidade sem necessitar do caso de uso “alterar disponibilidade laboratório()”

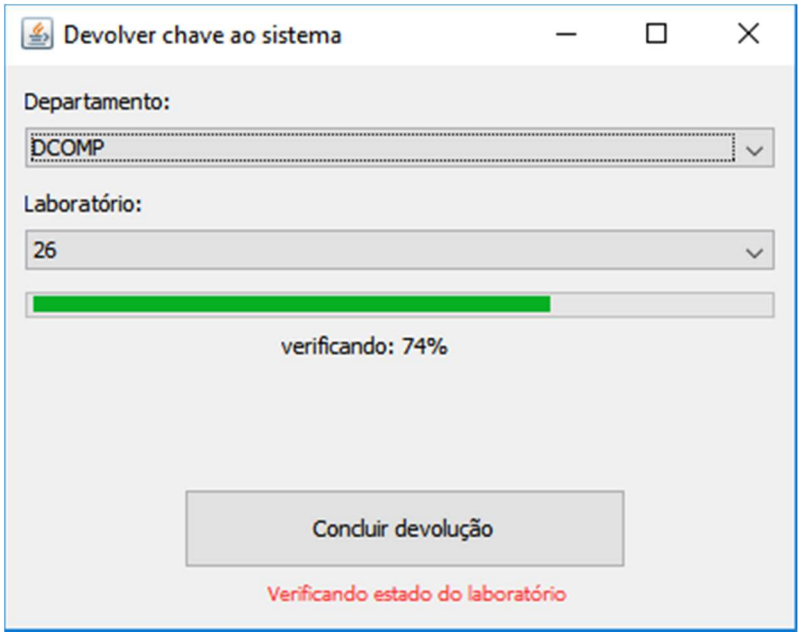
8. Para o fluxo Principal 9:

    1. O sistema deverá usar JInternalFrame como alternativa para substituir a JOptionPane, fazendo assim como que a mensagem seja mostrada para o Administrador.

9. Para o fluxo Principal 10:

    1. O fluxo deverá ser interrompido e finalizado.

Protótipos da Tela associado a este Caso de Uso:



- **Caso de Uso: Gerar Histórico()**

Descrição: Ao ser alterado o estado do laboratório, o sistema deverá gerar um histórico, de quem pegou a chave e quando devolveu;

Requisitos e Regras de Negócio: RF001, RF002, RF003, RF005, RF006, RFN002

Classes: Pessoa, Laboratório

Casos de uso: Entregar Chave(), Devolver chave()

Ator Primário: administrador;

Pré-Condição: O estado do laboratório deve ter sido alterado;

Pós-Condição: O histórico é gerado;

Fluxo Principal:

1. Após a entrega/devolução/manutenção o sistema deverá gerar uma conexão com o banco de dados;
2. O sistema deverá inserir as alterações que foram alteradas no sistema, dados do sistema e os dados das pessoas envolvidas em um pacote SQL, para isso deverá ser coletado o que ocorreu durante a entrega/devolução/manutenção.
3. O sistema deverá enviar um pacote SQL para o banco de dados, pedindo para que seja armazenado os dados na tabela Histórico.

Fluxo Alternativo:

1. Para o fluxo principal 1:

    1. O sistema deverá manter esses dados salvos no sistema até conseguir estabelecer uma conexão com o banco de dados.

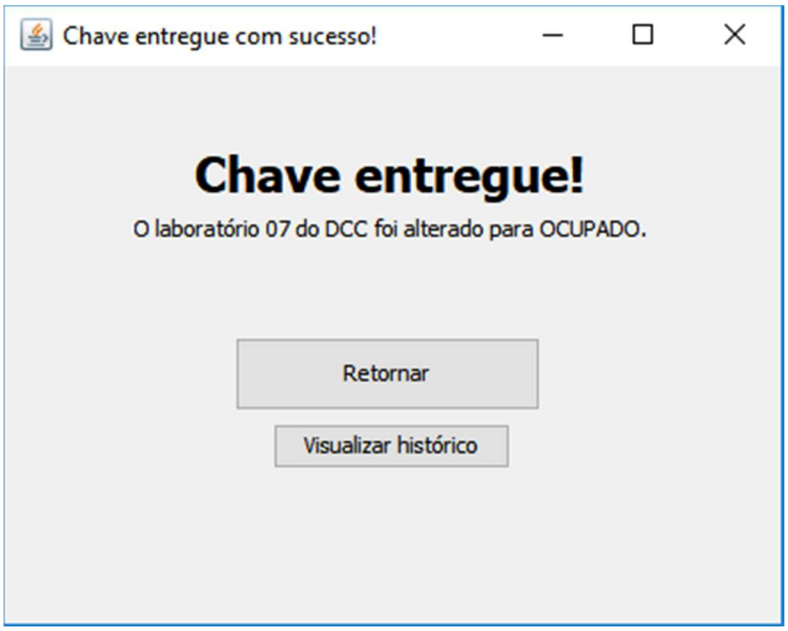
2. Para o fluxo principal 2:

1. Os dados deverão ser armazenados em um objeto de Relatório e este deverá ser utilizado para gerar o pacote SQL.

3. Para o fluxo principal 3:

    1. O sistema deverá manter esses dados salvos no sistema até conseguir estabelecer uma conexão com o banco de dados.

Protótipos da Tela associado a este Caso de Uso:



*Esse caso de uso não fica visível ao usuário, mas esse protótipo representa o momento em que ele será executado.

- **Caso de Uso: Verificar Histórico()**

Descrição: esse caso de uso serve para verificar o histórico, ou seja, realizar consultas no histórico;

Requisitos e Regras de Negócio: RF006, RNF002, RFN003

Classes: Gerenciar_historico, Relatório

Ator Primário: administrador;

Pré-Condição:Login() deve ter sido feito;

Pós-Condição: O sistema retornará o resultado da consulta;

Fluxo Principal:

1. O sistema deverá iniciar a janela “Procurar no histórico”, usando JInternalFrame do javax.swing, quando o administrador clicar no item de menu “Procurar” do menu “Histórico”
2. O sistema deverá iniciar uma conexão com o banco de dados

3. O sistema deverá preencher os JComboBox da janela “Procurar no histórico” com os dados de laboratórios e departamentos presentes no banco de dados
4. O administrador deverá informar o departamento e em sequência o laboratório
5. O sistema deverá então buscar os relatórios associados a esse laboratório
6. O sistema deverá preencher uma JTable com os dados localizados

Fluxo alternativo:

1. Para o fluxo Principal 1:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não conseguiu iniciar “Procurar no histórico” e se possível, por meio de Exceptions, informar o erro gerado.

    3. O fluxo deverá ser interrompido e finalizado.

2. Para o fluxo Principal 2:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não achar uma conexão com o banco de dados e se possível, por meio de SQL Exceptions, informar o erro gerado.

    3. O fluxo deverá ser interrompido e finalizado.

3. Para o fluxo Principal 3:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não conseguiu preencher os campos da janela e se possível, por meio de Exceptions, informar o erro gerado.

    3. O fluxo deverá ser interrompido e finalizado.

4. Para o fluxo Principal 4:

    1. O Sistema deverá manter a tela ativa independente do tempo que o administrador levar para informar o laboratório.

5. Para o fluxo Principal 5:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não conseguiu localizar o histórico referente ao laboratório escolhido e se possível, por meio de SQL Exceptions, informar o erro gerado.

    3. O fluxo deverá ser interrompido e finalizado.

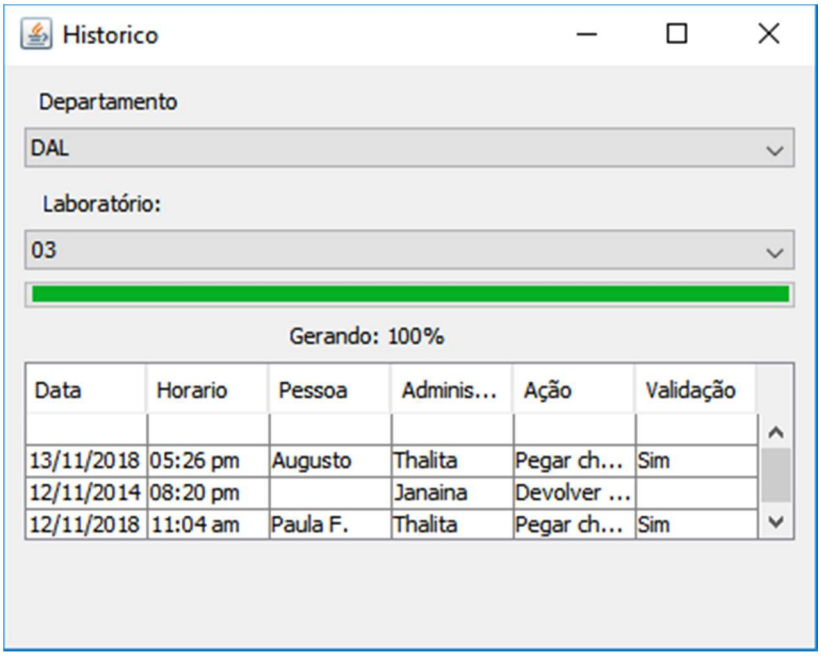
6. Para o fluxo Principal 6:

    1. O Sistema iniciará uma janela do JOptionPane

    2. O Sistema irá informar na janela que não conseguiu preencher a tabela e se possível, por meio de Exceptions, informar o erro gerado. Juntamente com essa janela teremos nela um botão recebem em texto onde os dados serão colocados em um campo de Texto de um JInternalFrame que será gerado quando o Administrador clicar nesse botão.

3. O fluxo deverá ser interrompido e finalizado.

Protótipos da Tela associado a este Caso de Uso:



- **Caso de Uso: Validar Assinatura()**

Descrição: O sistema irá verificar se a pessoa está cadastrada e autorizada, através da sua assinatura;

Requisitos e Regras de Negócio: RF001, RF005, RNF002, RNF003

Classes: Pessoa

Casos de uso: Entregar chave(), devolver chave()

Ator Primário: Pessoa;

Pré-Condição: Login() deve ter sido efetuado,A pessoa deve está cadastrada no sistema;

Pós-Condição: A assinatura será declarada como válida;

Fluxo Principal:

1. O sistema deverá receber uma variável do tipo String, que possivelmente é uma assinatura
2. O sistema deverá criptografar essa variável utilizando o mesmo padrão do banco de dados
3. O sistema deverá criar uma conexão com o banco de dados (caso não exista nenhuma ativa)
4. O sistema enviará uma consulta SQL buscando se aquela assinatura está no sistema
5. O sistema irá retornar que a assinatura é válida

Fluxo Alternativo:

1. Para os Fluxos Principais 1,2,3:
    1. O Sistema iniciará uma janela do JOptionPane

2. O Sistema irá informar na janela que não conseguiu verificar a assinatura recebida e se possível, por meio de SQL Exceptions, informar o erro gerado.

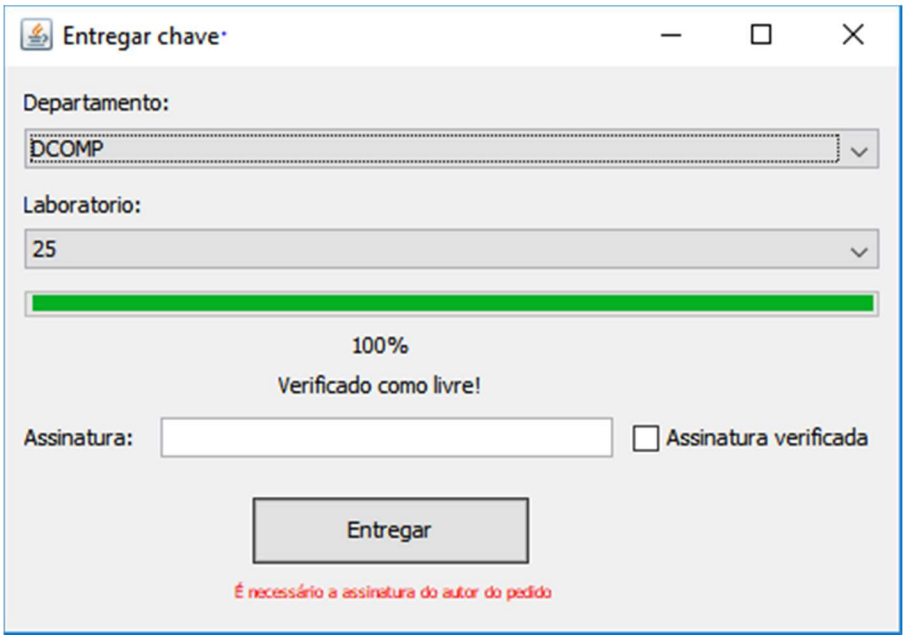
3. O fluxo deverá ser interrompido e finalizado.

2. Para o Fluxo principal 4:

1. O sistema deverá retornar que a assinatura não é válida.

3. Para o Fluxo principal 5:
4. O fluxo deverá ser interrompido e reiniciado.

Protótipos da Tela associado a este Caso de Uso:



- **Caso de Uso: Verificar Departamento()**

Descrição: o sistema irá verificar se o professor é do mesmo departamento do laboratório.

Requisitos e Regras de Negócio: RF001, RF004, RNF002, RNF003

Classes: Departamento, Professor

Casos de uso: Entregar chave()

Ator Primário: administrador;

Pré-Condição: login() deve ter sido efetuado, a pessoa deve está cadastrada no sistema;

Pós-Condição: A pessoa receberá terá autorização para seguir o procedimento de pegar chave

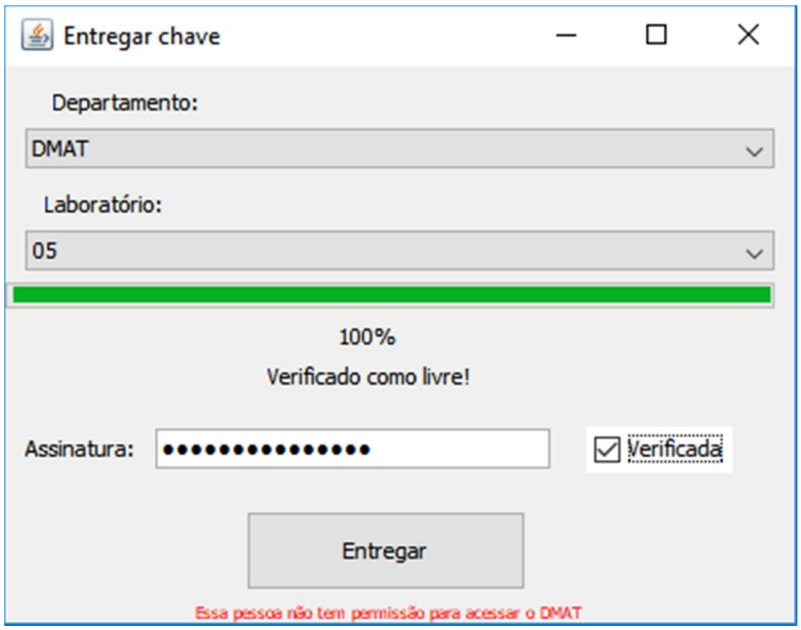
Fluxo Principal:

1. O sistema deverá receber uma pessoa e o laboratório que esta deseja;
2. O sistema irá verificar se ambos estão associados ao mesmo departamento;
3. O acesso será permitido;

Fluxo Alternativo:

1. O sistema deverá informar que não é possível verificar o departamento utilizando uma janela da JOptionPane e ,além disso, retornar que o acesso deve ser negado.

Protótipos da Tela associado a este Caso de Uso:



- **Caso de uso: Manutenção()**

Descrição: O sistema irá verificar se o laboratório receberá uma manutenção ou se deverá encerrar uma manutenção existente.

Requisitos e Regras de Negócio: RF001, RF004, RNF002, RNF003

Classes: Laboratório

Casos de uso: Alterar estado laboratório()

Ator Primário: administrador;

Pré-Condição: login() deve ter sido efetuado

Pós-Condição: O laboratório terá sua disponibilidade alterada

Fluxo Principal:

1. Quando o administrador clicar no item de menu “Manutenção” do menu “Laboratório”, o sistema deverá iniciar a janela “Manutenção de Laboratório” utilizando a classe JInternalFrame do pacote javax.swing.
2. O sistema deverá iniciar uma conexão com o banco de dados.

3. O sistema deverá preencher os JComboBox da janela com os dados de Departamentos e Laboratórios presentes no banco de dados
4. O administrador deverá interagir com o sistema para informar o laboratório
5. O sistema deverá informar a disponibilidade desse laboratório e ,ao mesmo tempo, verificar se o laboratório está com as disponibilidades “Em manutenção” ou “Disponível”, caso esteja “ocupado” ou “em limpeza” o fluxo deverá seguir para o alternativo 7.
6. O sistema irá adicionar um botão na tela baseado na disponibilidade do laboratório.
    a. Caso esteja “disponível”, será adicionado o botão “Nova manutenção”.
    b. caso esteja “Em manutenção” será adicionado o botão “Encerrar manutenção”
7. Quando clicado, o botão irá chamar uma JOptionPane confirmando a ação(Iniciar manutenção ou finalizar) e assim que confirmado irá chamar o caso de uso “Alterar disponibilidade” passando o laboratório e a disponibilidade indicada pelo botão
    a. o botão “Nova manutenção” passa a disponibilidade “Em manutenção”
    b. o botão “Encerrar manutenção” passa a disponibilidade “disponível”
8. O sistema deverá confirmar que atualizou a disponibilidade, para isso será utilizada um JOptionPane e assim que o administrador confirmar como lida a mensagem da JOptionPane o fluxo deverá reiniciar.

Fluxo Alternativo:

1. Para o fluxo principal 1:

    1. O sistema deverá informar por meio de uma janela JOptionPane que não foi possível iniciar o caso de uso, interrompendo o fluxo e encerrando o caso de uso.

2. Para o fluxo principal 2:

    1. O sistema deverá informar por meio de uma janela JOptionPane que não foi possível se comunicar com o banco de dados, interrompendo o fluxo e encerrando o caso de uso(Finalizando tudo que já foi iniciado).

3. Para o fluxo principal 3,5,6,7:

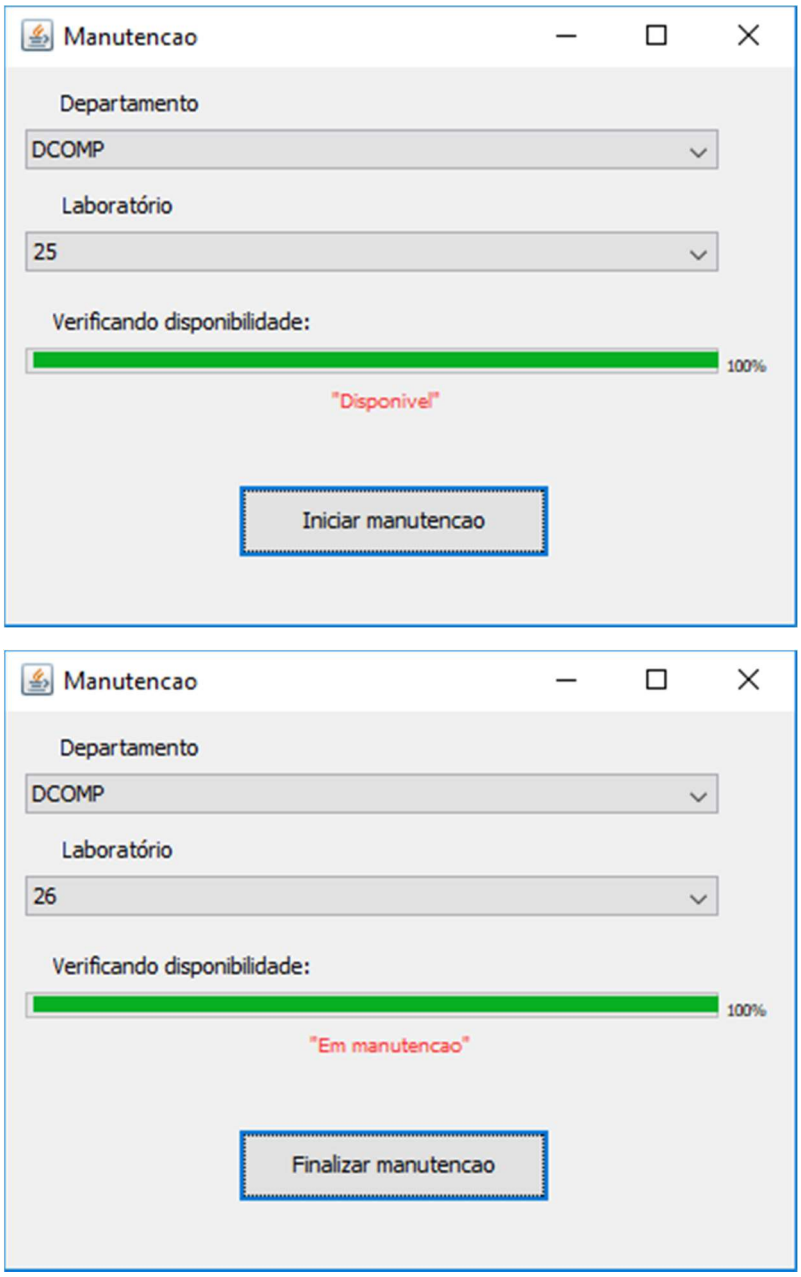
    1. O sistema deverá informar por meio de uma janela JOptionPane o erro que foi capturado pela Exception, interrompendo o fluxo e encerrando o caso de uso(Finalizando tudo que já foi iniciado).

4. Para o fluxo principal 4:

    1. O sistema deverá manter-se ativo para receber os dados, independentemente do tempo que o administrador leve para informar o laboratório

5. Para o fluxo principal 8:
6. O sistema deverá finalizar a janela “Manutenção de laboratório” aberta e iniciar uma nova, nesta deverá ser fixado um alerta de “a disponibilidade do laboratório ** (deve se informado qual laboratório) do *****(Aqui deve ser informado o departamento do laboratório) foi alterada para ****** (A disponibilidade que esse laboratório recebeu)!”
7. O sistema deverá abrir uma janela JOptionPane informando que esse laboratório deve ser desocupado para que se tenha início a manutenção.

Protótipos da Tela associado a este Caso de Uso:



- **Caso de uso: Gerar Header()**

Descrição: Esse caso de uso serve para gerar um identificador numérico para uma determinada data.

Requisitos e Regras de Negócio: RF006, RNF001, RNF003

Classes: Gerenciar_historico, Relatório

Ator Primário: Nenhum

Pré-Condição: Login() deve ter sido feito, uma data e horário deve ter sido identificados;

Pós-Condição: Uma data terá um valor numérico associado;

Fluxo Principal:

1. O sistema irá receber uma data/hora.
2. Deverá ser iniciada uma conexão com o banco de dados
3. O sistema deverá inserir a data/hora no banco e o banco deverá gerar um id pelo sistema de auto-increment
4. O sistema deverá retornar a confirmação de que conseguiu inserir no banco

Fluxo Alternativo:

1. Para o fluxo principal 1:

   1.1 O sistema deverá retornar que a data é invalida

2. Para o fluxo principal 2:

   2.1 O sistema deverá retornar que não foi possível estabelecer uma conexão com o banco de dados

3. Para o fluxo principal 3:

   3.1 O sistema deverá retornar que não foi possível inserir a data no sistema

4. Para o fluxo principal 4:

   4.1 O sistema deverá iniciar uma janela JOptionPane

   4.2 O sistema deverá informar que é necessário reiniciar o sistema

Protótipos de Tela:

-Esse caso de uso é oculto para o usuário, não tendo protótipos de telas que o defina.

- **caso de uso: Buscar Header()**

Descrição: Esse caso de uso serve para retornar data(s) e/ou identificador(es) dependendo do Pacote SQL que for passado.

Requisitos e Regras de Negócio: RF006, RNF001, RNF003

Classes: Gerenciar_historico, Relatório

Ator Primário: Nenhum

Pré-Condição: Login() deve ter sido feito, Tabela ‘Header’ no banco de dados não deve está vazia;

Pós-Condição: Uma ou mais datas serão retornadas com seus respectivos identificadores númericos;

Fluxo Principal:

5. O sistema irá receber uma String SQL criptografada
6. Deverá ser iniciada uma conexão com o banco de dados
7. O sistema deverá descriptografar e executar o SQL
8. O sistema deverá retornar o resultado do SQL

Fluxo Alternativo:

5. Para o fluxo principal 1:

   1.1 O sistema deverá retornar que o parâmetro é inválido

6. Para o fluxo principal 2:

   2.1 O sistema deverá retornar que não foi possível estabelecer uma conexão com o banco de dados

7. Para o fluxo principal 3:

   3.1 O sistema deverá retornar que não foi possível executar o SQL no banco

8. Para o fluxo principal 4:

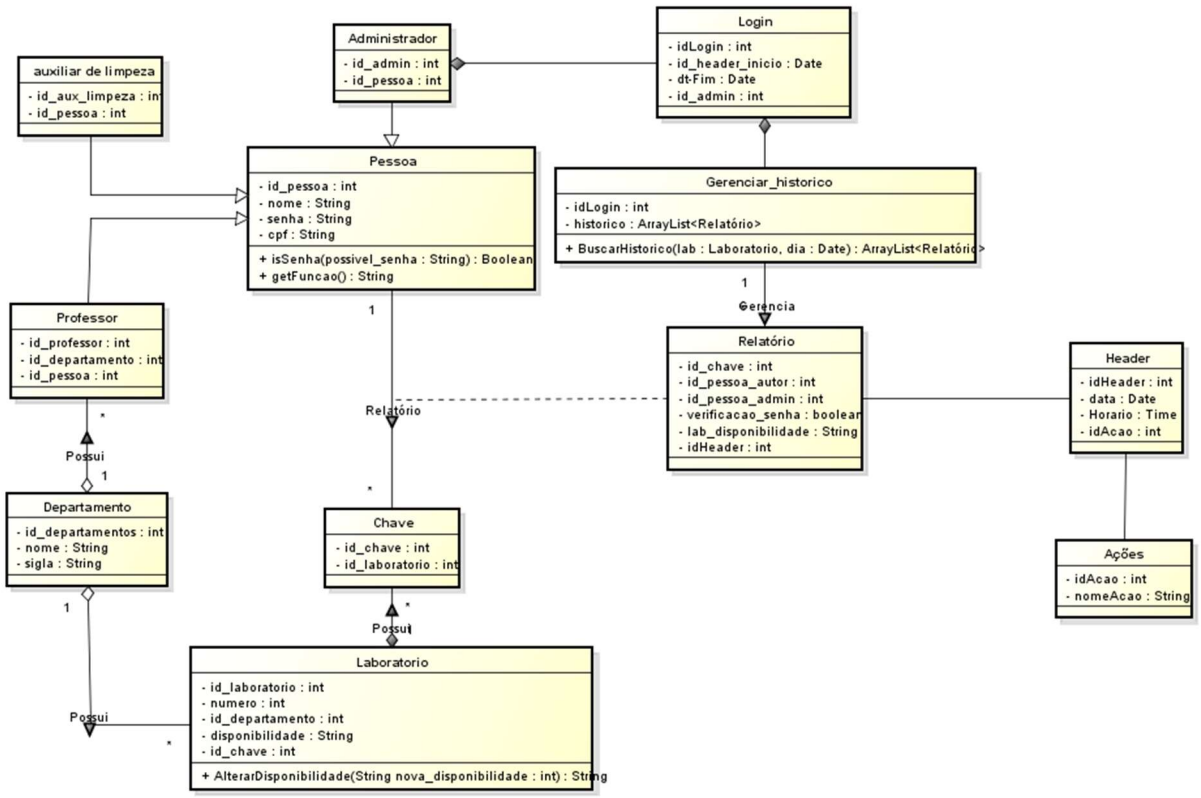
   4.1 O sistema deverá iniciar uma janela JOptionPane

   4.2 Deverá informar então que é necessário reiniciar o sistema

Protótipos de Tela:

-Esse caso de uso é oculto para o usuário, não tendo protótipos de telas que o defina.

6. Modelagem de domínio



- **Pessoa**

  Finalidade: Essa entidade representa as pessoas, em geral, que irão interagir com o sistema de diversas maneiras.

  Atributos:

  id_pessoa: Identificador único do tipo inteiro, usado para distinguir cadastros semelhantes.

  nome: Substantivo próprio da pessoa do tipo String.

  senha: Atributo do tipo String, deve ser único e usado uma criptografia para garantir a integridade do sistema, a senha é usada para confirmar a identidade de uma pessoa.

  cpf: Atributo do tipo String, deve ser único e válido, usado para garantir a existência física da pessoa.

  Metodos:

  isSenha(String possivel_senha)

  definição: Devido a senha estar criptografada, um simples getSenha não seria suficiente e acabaria ferindo o princípio do encapsulamento. Dessa forma, será utilizado o método isSenha que deverá verificar se o valor passado é igual a senha salva da pessoa (OBS* O método de criptografia ainda não foi definido), retornando assim apenas uma confirmação de que o valor confere com a senha da pessoa.

retorno: O retorno será um valor do tipo **boolean**

possíveis resultados de retorno:

1.True, caso o parâmetro passado seja igual a senha atual da pessoa.

2.False, caso o parâmetro passado seja diferente da senha atual da pessoa.

3. Null, caso tenha falhado a verificação

getFuncao()

definição: No sistema, uma pessoa é a generalização de professores, auxiliares de limpeza e dos administradores, de forma que uma pessoa pode assumir uma dentre essas 3 funções. O método getFuncao irá buscar e retornará qual é a função da pessoa no sistema.

retorno: O retorno será uma variável do tipo **String**

possíveis resultados de retorno:

1. 'professor', caso a pessoa esteja associada ao registro de um professor.

2. 'auxiliar', caso a pessoa esteja associada ao registro de um auxiliar

3. 'administrador', caso a pessoa esteja associada ao registro de um administrador.

4. 'null', caso a função não tenha sido encontrada.

Relacionamento: Associação com a entidade chave.

Multiplicidade: Uma Pessoa possui um número indeterminado Chaves.

- **Auxiliar de limpeza**

Finalidade: Essa entidade representa as pessoas, responsáveis pela higiene dos laboratórios, que irão interagir com o sistema pegando e devolvendo chaves de laboratórios.

Atributos:

id_aux_limpeza: Identificador único do tipo inteiro, usado para distinguir cadastros semelhantes.

id_pessoa: Chave estrangeira do tipo inteiro, usado para associar o auxiliar com uma pessoa.

Métodos:

Essa entidade apresenta apenas os métodos get.

Relacionamento: Generalização de pessoa;

- **Administrador**

Finalidade: Essa entidade representa as pessoas, responsáveis pela distribuição e armazenamento das chaves dos laboratórios, que irão interagir com o sistema entregando e recebendo as chaves, assim como analisando o estado e o histórico de um determinado laboratório.

Atributos:

id_admin: Identificador único do tipo inteiro, usado para distinguir cadastros semelhantes.

id_pessoa: Chave estrangeira do tipo inteiro, usado para associar o auxiliar com uma pessoa.

Métodos:

Essa entidade apresenta apenas os métodos get.

Relacionamento: Generalização de pessoa e composição com a entidade Gerenciar_historico.

Multiplicidade: Um Administrador possui um Gerenciador_historico.

- **Professor**

Finalidade: Essa entidade representa as pessoas, responsáveis por utilizar os laboratórios para realizar aulas,exercícios ou outras atividades do gênero, que irão interagir com o sistema pegando e devolvendo chaves de laboratórios.

Atributos:

id_professor: Identificador único do tipo inteiro, usado para distinguir cadastros semelhantes.

id_departamento: Chave estrangeira do tipo inteiro, usada para associar o professor com um departamento.

id_pessoa: Chave estrangeira do tipo inteiro, usada para associar o professor com uma pessoa.

Métodos:

Essa entidade apresenta apenas os métodos get.

Relacionamento: Generalização de pessoa e agregação com Departamento;

Multiplicidade: Vários Professores fazem parte de um Departamento.

- **Laboratório**

Finalidade: Essa entidade representa o espaço físico destinado para realização de experimentos, aulas, trabalhos, entre outras atividades. Esse espaço será gerenciado pelo sistema, onde o sistema ficará responsável por sua chave, assim como por disponibilizá-lo ou identificá-lo como ocupado/ em manutenção.

Atributos:

id_laboratorio: Atributo chave primária do tipo inteiro, responsável por identificar e diferenciar os cadastros dos laboratórios.

numero: Atributo do tipo inteiro, para identificar o número do laboratório.

disponibilidade: Atributo do tipo String, responsável por dizer o estado do laboratório.

id_departamento: Atributo do tipo inteiro, responsável por determinar de qual departamento é o laboratório.

id_chave: Atributo do tipo inteiro, responsável por identificar a chave do laboratório.

Métodos:

alterarDisponibilidade(String nova_disponibilidade)

Descrição: Método irá alterar a disponibilidade do laboratório de acordo com o seu parâmetro.

retorno: O retorno será do tipo String.

Relacionamento: Agregação com um departamento e tem uma chave relacionada por composição.

Multiplicidade: Vários Laboratório fazem parte de um Departamento e contém várias Chaves.

- **Departamento**

Finalidade: Essa entidade representa os Departamento, responsável por gerenciar e identificar os laboratórios e pessoas que a compõem.

Atributos:

id_departamento: Atributo chave primária do tipo inteiro, responsável por identificar e diferenciar os cadastros dos Departamentos.

nome: Atributo do tipo String, responsável por dizer o nome do departamento.

sigla: Atributo do tipo String, responsável por dizer a sigla do departamento.

Métodos:

Essa entidade apresenta apenas os métodos get.

Relacionamento: Tem agregado as entidades Professor e Laboratório;

Multiplicidade: Departamento possui vários Professores e Laboratórios.

- **Chave**

Finalidade: Essa entidade representa as chaves, responsável por abrir o laboratório.

Atributos:

id_chave: Atributo chave primária do tipo inteiro, responsável por identificar e diferenciar os cadastros das chaves.

id_laboratorio: Atributo do tipo inteiro, responsável por identificar de qual laboratório é a chave.

Métodos:

Essa entidade apresenta apenas os métodos get.

Relacionamento: Tem composição com laboratório e tem uma associação com pessoa.

Multiplicidade: Existem varias Chaves para um Laboratório.

- **Relatório**

Finalidade: Essa entidade é responsável pelo armazenamento de toda movimentação de chave que acontecer no sistema, de modo que armazenará informações referentes ao horário, laboratório e pessoas envolvidas em uma movimentação.

Atributos:

id_chave: Atributo do tipo inteiro, responsável por identificar a chave.

id_pessoa_autor: Atributo do tipo inteiro, responsável por identificar a pessoa.

id_pessoa_admin: Atributo do tipo inteiro, responsável por identificar o administrador.

acao: Atributo do tipo String, responsável por dizer a ação que será realizada.

data: Atributo do tipo Date, responsável por registrar a data do relatório.

verificação_senha: Atributo do tipo Boolean, responsável por verificar a senha.

lab_disponibilidade: Atributo do tipo String, responsável por dizer a disponibilidade do laboratório.

Métodos:

Essa entidade apresenta apenas os métodos get.

Relacionamento: Entidade associativa criada a partir da associação de pessoa com chave e essa entidade também está associada com a entidade Gerenciar_historico.

Multiplicidade: Existem vários Relatórios para um Gerenciar_historico.

- **Gerenciar_historico**

Finalidade: Essa entidade é responsável pelo gerenciamento dos relatórios gerados durante todo o período de execução do sistema, de modo que ela consiga buscar e organizar relatórios baseados em um laboratório e/ou período.

Atributos:

Historico: Atributo do tipo ArrayList<Historico>, responsável por conter o histórico dos laboratórios.

Métodos:

BuscarHistorico(laboratório : int, data : date)

descrição: Esse método irá reunir uma coleção de relatórios de um laboratório usando como parâmetro um período de tempo determinado ou indeterminado(Caso não seja passado a data no parâmetro do método).

retorno: O retorno será um arraylist do tipo Relatório.

possíveis casos de retorno:

1. Relatórios de um dia: Todos os relatórios gerados em um dia especifico que estão associados ao laboratório determinado.

2. Relatórios de todo o período: Todos os relatórios gerados que estão associados ao laboratório determinado, independente da data em que esse relatório foi gerado. Esse retorno ocorrerá quando não for passado uma data específica.

3. Null: Caso ocorra algum erro durante a execução, exemplo: laboratório não localizado, relatórios não localizados, data inválida, etc.

Relacionamento: Composição com a entidade Administrador e Associação com a entidade Relatório.

Multiplicidade: Um Gerenciar_historico para vários Relatórios.

- **Header**

Finalidade: Essa entidade é responsável por associar uma data à um identificador do tipo numérico.

Atributos:

Historico: Atributo do tipo ArrayList<Historico>, responsável por conter o histórico dos laboratórios.

IdHeader: Atributo do tipo Integer, chave primária responsável por identificar um data/horário no sistema.

Data: Atributo do tipo date.

Horário: Atributo do tipo time;

Ação: Chave estrangeira que associa a ação que ocorreu no respectivo header.

Métodos:

BuscarHistorico(laboratório : int, data : date)

descrição: Esse método irá reunir uma coleção de relatórios de um laboratório usando como parâmetro um período de tempo determinado ou indeterminado(Caso não seja passado a data no parâmetro do método).

retorno: O retorno será um arraylist do tipo Relatório.

possíveis casos de retorno:

1. Relatórios de um dia: Todos os relatórios gerados em um dia especifico que estão associados ao laboratório determinado.

2. Relatórios de todo o período: Todos os relatórios gerados que estão associados ao laboratório determinado, independente da data em que esse relatório foi gerado. Esse retorno ocorrerá quando não for passado uma data específica.

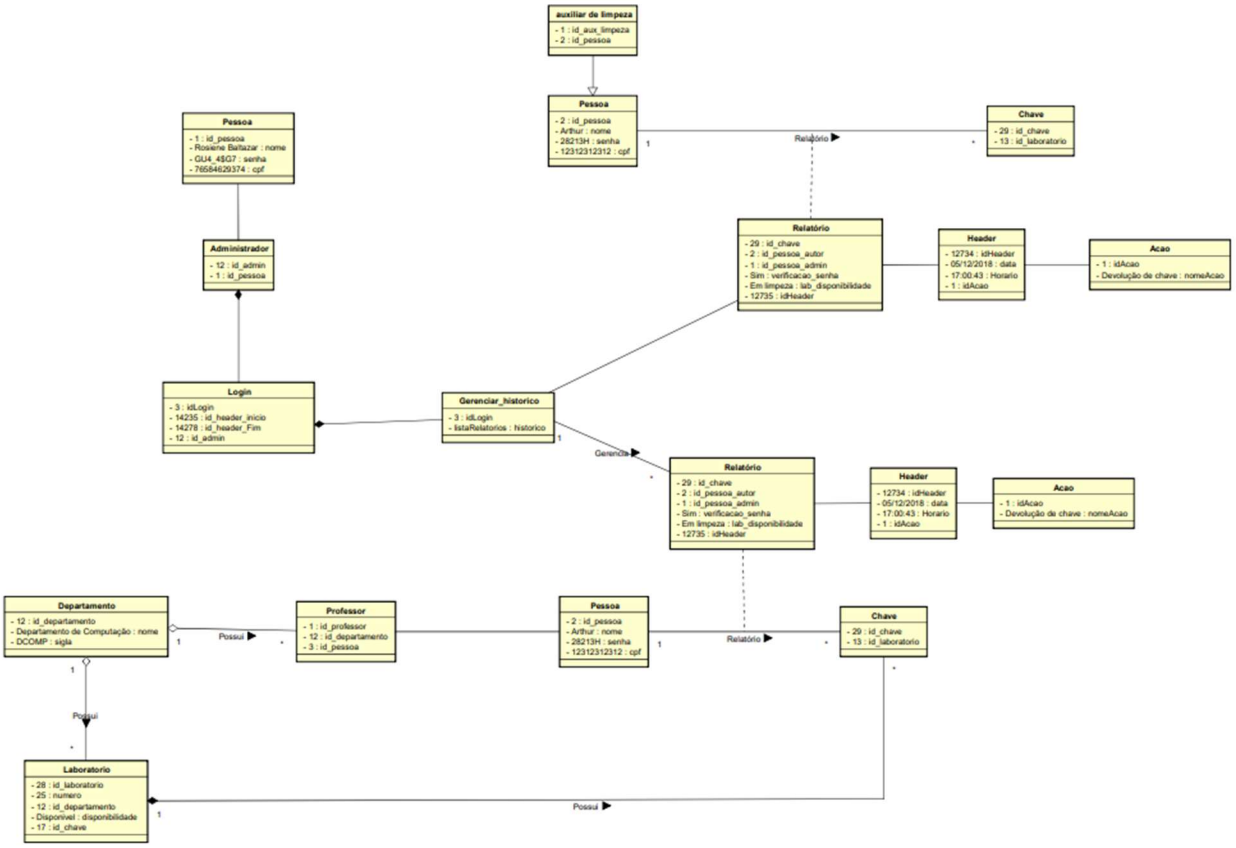
3. Null: Caso ocorra algum erro durante a execução, exemplo: laboratório não localizado, relatórios não localizados, data inválida, etc.

Relacionamento: Composição com a entidade Administrador e Associação com a entidade Relatório.

Multiplicidade: Um Gerenciar_historico para vários Relatórios.

6.1 Diagrama de Objeto

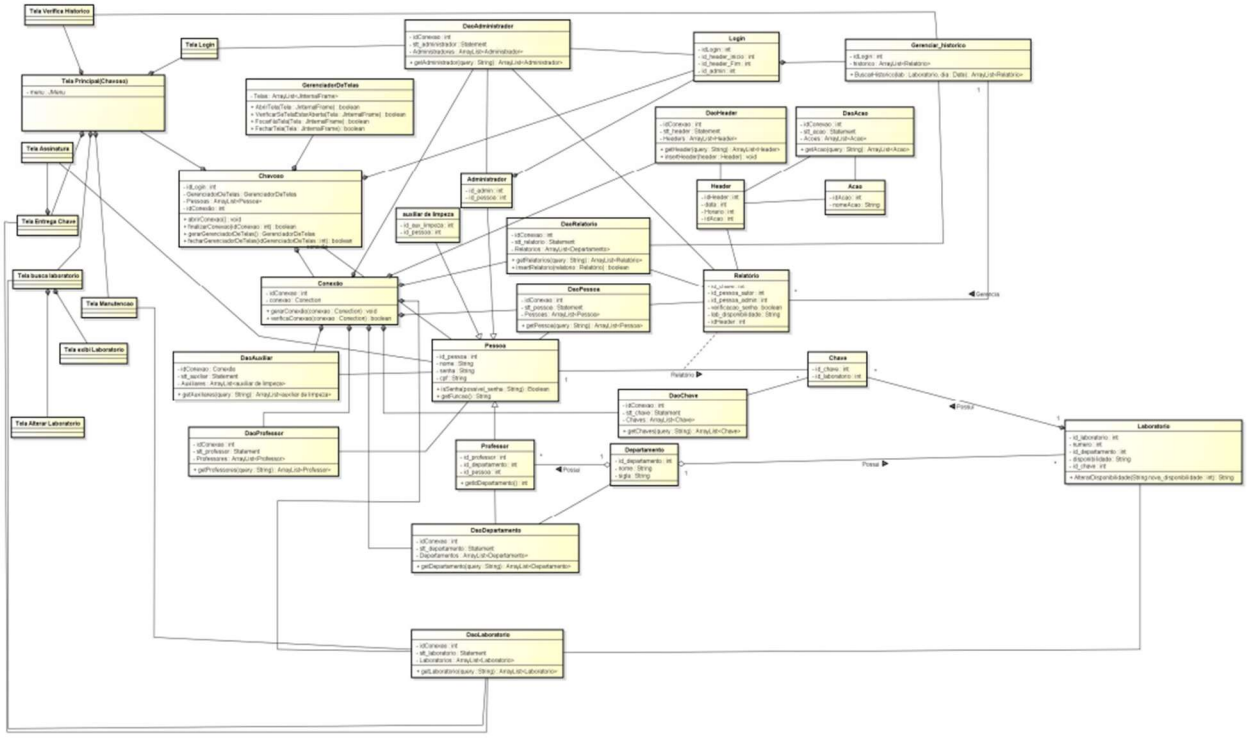
Durante a execução do sistema teremos diversas relações entre as entidades.



*Vale ressaltar que essa são entidades exemplos e que isso é apenas uma simulação de execução

Nesse diagrama é possível percebemos como essas relações ocorrem. Detalhadamente, temos que cada administrador, professor ou auxiliar da limpeza terá uma entidade pessoa, única, associada a si. Essa pessoa ao se relacionar com uma chave irá gerar um relatório e todos os relatórios serão associado ao gerenciar_historico que por sua vez está ligado à login por meio de uma composição.

7. Diagrama de classe



Class Chavoso:

Descrição: A classe principal do sistema, é responsável por gerenciar funcionalidades que são transparentes para o usuário e ao mesmo tempo garante que funcionalidades visíveis fiquem acessíveis ao usuário.

Classes relacionadas: GerenciadorDeTelas, Conexão, Login, Pessoa.

- Métodos:
    - Interação(Iniciar ou finalizar conexão) com o BD(abrirConexao() // finalizarConexao(idConexao));
    - Interagir com o gerenciador de tela para poder manusear as mesmas(gerarGerenciadorDeTelas()//fecharGerenciadorDeTelas(idGerenciadorDeTelas));

Class Conexão:

Descrição: Responsável pela geração e verificação da conexão do sistema com o BD.

Classes relacionadas: Chavoso e todas as classes DAO's.

- Métodos:
    - Gerar a conexão com o BD(gerarConexão(conexao : Conection));
    - Fazer a verificação dessa conexão(verificaConexao(conexao : Conection);

Class GerenciadorDeTelas:

Descrição: Responsável por toda versatilidade no manuseamento e interação do usuário com as telas(Interfaces).

Classes relacionadas: Chavoso.

- Métodos:
    - Interações com a tela(AbrirTela(Tela : JinternalFrame) // VerificarSeTelaEstarAberta(Tela : JInternalFrame) // FocarNaTela(Tela : JInternalFrame) // FecharTela(Tela : JInternalFrame));

Class Pessoa:

Descrição: Responsável pelos dados de uma pessoa(Nome , senha , CPF) ,sendo a generalização de professor, auxiliar da limpeza e administrador.

Classes relacionadas: Chavoso, professor, auxiliar da limpeza, administrador, DaoPessoa, DaoProfessor, DaoAuxiliar, Chave, Relatorio.

- Funções:
    - “Pegar” a função da pessoa cadastrada(getFuncao());
    - Verificar a senha cadastrada(isSenha(possivel_senha : String));

Class Professor:

Descrição: Responsável por receber os dados de um professor designado da classe pessoa e receber dados exclusivos dessa classe como um departamento.

Classes relacionadas: Pessoa, DaoDepartamento, Departamento.

- Métodos:
    - Cadastrar um departamento de acordo com o professor(getIdDepartamento());

Class auxiliar de limpeza:

Descrição: Responsável por receber os dados de um auxiliar de limpeza designado da classe pessoa e receber dados exclusivos dessa classe como o id do auxiliar.

Classes relacionadas: Pessoa.

- Métodos:
    - Inexistente;

Class Administrador:

Descrição: Responsável por receber os dados de um administrador designado da classe pessoa e receber dados exclusivos dessa classe como o id do administrador.

Classes relacionadas: Pessoa, DaoAdministrador, Login.

- Métodos:
    - Inexistente;

Class Departamento:

Descrição: Responsável por receber todos os dados de um departamento como nome e sigla.

Classes relacionadas: Professor, Laboratório;

- Métodos:
    - Inexistente;

Class Laboratorio:

Descrição: Responsável por receber todos os dados de um laboratório como número, disponibilidade.

Classes relacionadas: DaoLaboratorio, Departamento, Chave.

- Métodos:
    - Alterar o status de um laboratório(AlterarDisponibilidade(String nova_disponibilidade : int));

Class Login:

Descrição: Responsável por receber os dados para a efetuação do login por um administrador.

Classes relacionadas: Gerenciar_historico, DaoAdministrador, Chavoso, Administrador.

- Métodos:
    - Inexistente;

Class Relatório:

Descrição: Gerado todo vez que ocorre uma entrega ou recebimento de uma chave. Passando as alterações necessária para serem armazenadas.

Classes relacionadas: DaoAdministrador, DaoRelatorio, Daopessoa, Gerenciar_historico e também com a interação da classe pessoa e a chave.

- Métodos:
    - Inexistente;

Class Gerenciar_historico:

Descrição: Responsável por gerenciar todos os históricos do sistema.

Classes relacionadas: DaoRelatorio, Relatorio, Login.

- Métodos:
    - Permitir a busca de um determinado histórico(BuscarHistorico(lab : Laboratorio, dia : Date);

Class Header:

Descrição: Responsável por transformar um conjunto data/hora em um valor numérico(idHeader) de modo que ações que ocorrem no mesmo horário gerem apenas a repetição de um valor numérico(caso repetisse o conjunto data/hora o banco de dados teria um tamanho consideravelmente maior e um trabalho mais complexo para a implementação pois teria operações com data/hora) .

Classes relacionadas: Acao, DaoHeader, DaoAcao, Relatorio.

- Métodos:
    - Inexistente;

Class Acao:

Descrição: Responsável por gerenciar determinada a ser ação realizada pelo sistema.

Classes relacionadas: DaoAcao, Header.

- Funções:
    - Inexistente;

Class Chave:

Descrição: Responsável por ser a representação lógica de um objeto físico que é utilizado pelo sistema.

Classes relacionadas: Laboratorio, DaoChave, Pessoa, Relatorio.

- Métodos:
    - Inexistente;

DAO's

As classes com o prefixo "Dao" representam classes que realizam ações DML e DQL em conjunto ao banco de dados.

Class DaoAuxiliar:

Descrição: Permite buscar os dados dos auxiliares da limpeza no banco de dados;

Classes relacionadas: Conexão, Pessoa.

- Métodos:
    - Selecionar os dados no BD(getAuxiliares(query : String));

Class DaoProfessor:

Descrição: Permite buscar os dados dos professores no banco de dados;

Classes relacionadas: Conexão, Pessoa.

- Métodos:
    - Selecionar os dados no BD(getProfessores(query : String));

Class DaoAdministrador:

Descrição: Permite buscar os dados dos administradores no banco de dados;

Classes relacionadas: Conexão, Administrador, Relatorio.

- Métodos:
    - Selecionar os dados no BD(getAdministrador(query : String));

Class DaoPessoa:

Descrição: Permite buscar os dados das pessoas no banco de dados;

Classes relacionadas: Conexão, Pessoa, Relatorio.

- Métodos:
    - Selecionar os dados no BD(getPessoa(query : String));

Class DaoRelatorio:

Descrição: Permite buscar e inserir novos relatórios no banco de dados;

Classes relacionadas: Conexão, Gerenciar_historico, Relatorio.

- Métodos:
    - Selecionar os dados no BD(getRelatorios(query : String));
    - Inserir novos relatórios no BD(insertRelatorio(relatorio : Relatorio));

Class DaoDepartamento:

Descrição: Permite buscar os dados dos departamentos no banco de dados;

Classes relacionadas: Conexão, Professor, Departamento.

- Métodos:
    - Selecionar os dados no BD(getDepartamento(query : String));

Class DaoLaboratorio:

Descrição: Permite buscar os dados dos laboratórios no banco de dados;

Classes relacionadas: Conexão, Laboratorio.

- Métodos:
    - Selecionar os dados no BD(getLaboratorio(query : String));

Class DaoChave:

Descrição: Permite buscar os dados das chaves no banco de dados;

Classes relacionadas: Conexão, Chave.

- Métodos:
    - Selecionar os dados no BD(getChaves(query : String));

Class DaoHeader:

Descrição: Permite buscar e inserir headers no banco de dados;

Classes relacionadas: Conexão, Header.

- Métodos:
    - Selecionar os dados no BD(getHeader(query : String));
    - Inserir novos headers no BD(insertHeader(header : Header));

Class DaoAcao:

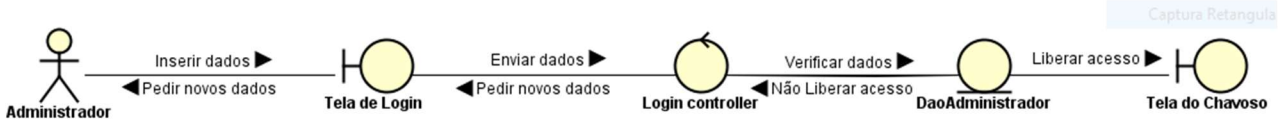
Descrição: Permite buscar os dados dos headers no banco de dados;

Classes relacionadas: Conexão, Header, Acao.

- Métodos:
    - Selecionar os dados no BD(getAcao(query : String));

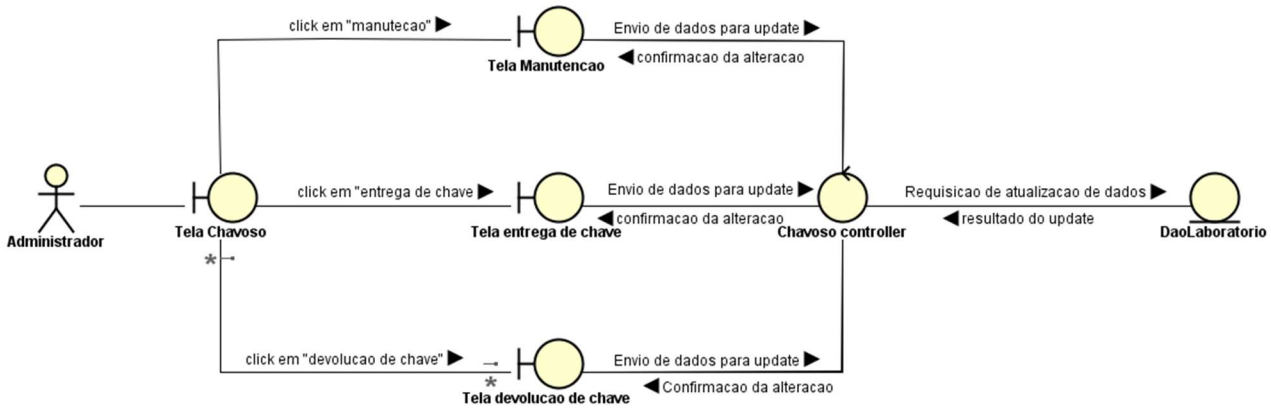
8. Diagrama de robustez

8.1 Login()

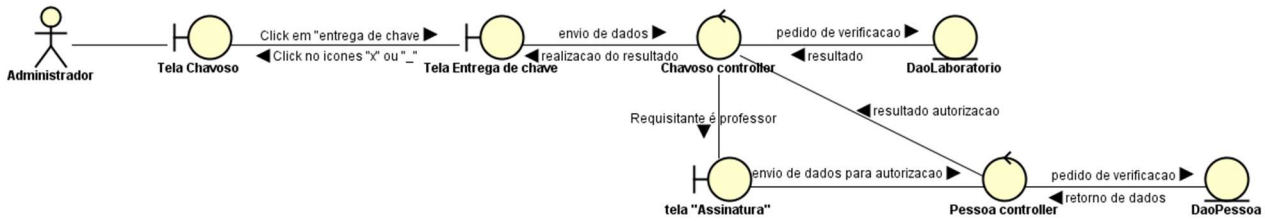


8.2 Disponibilidade()

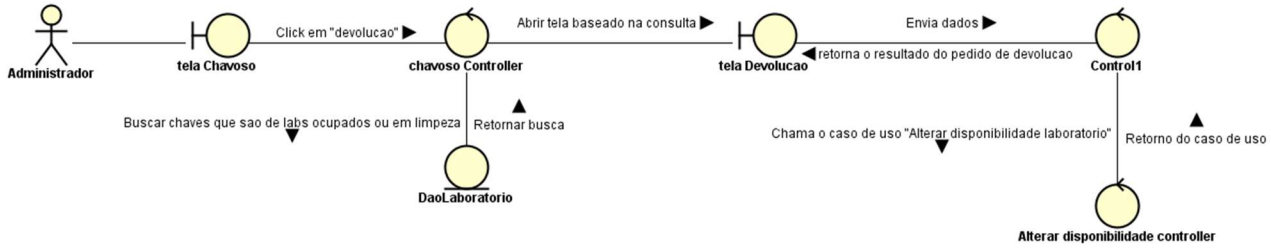
8.3 Alterar disponibilidade()



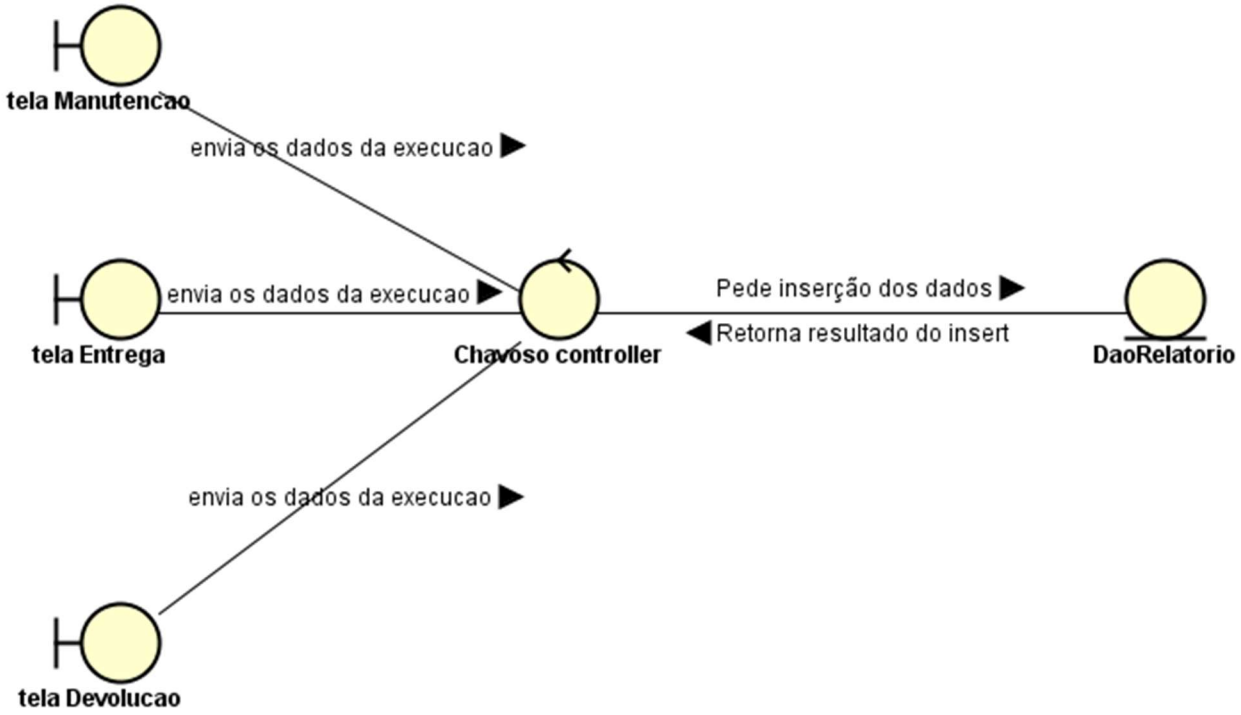
8.4 Entregar chave()



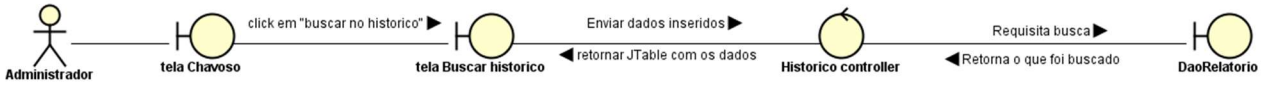
8.5 Devolver chave()



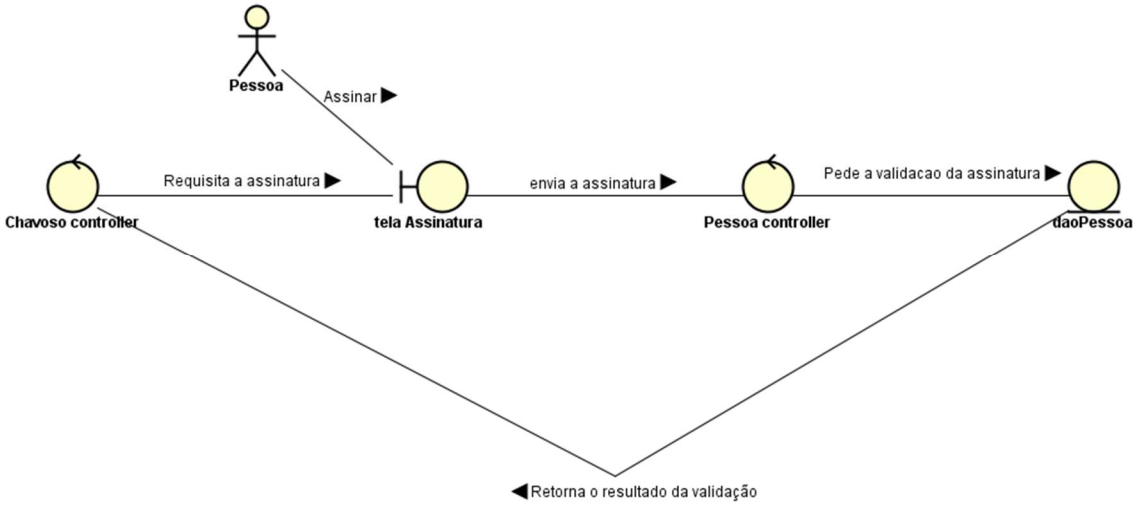
8.6 Gerar histórico()



8.7 Verificar histórico()

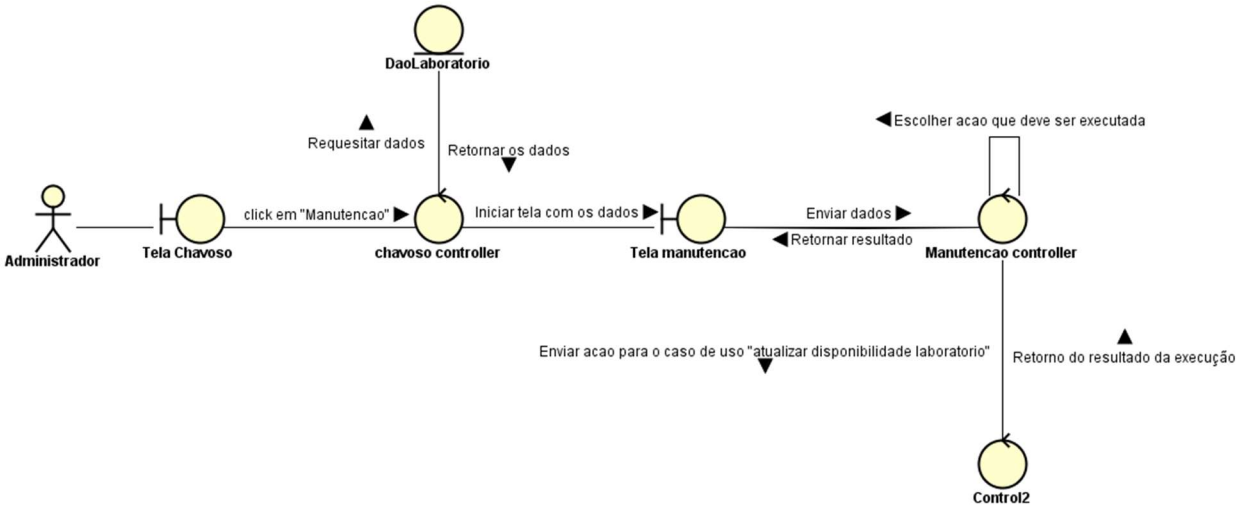


8.8 Validar assinatura()

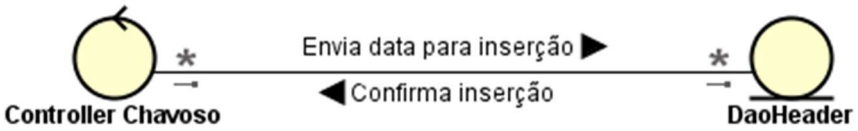


8.9 Verificar departamento()

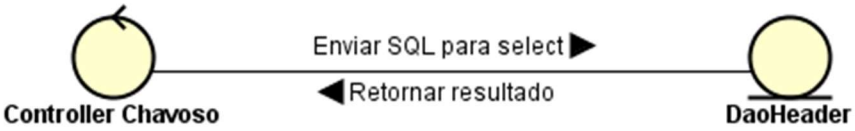
8.10 Manutenção()



8.11 Gerar Header()

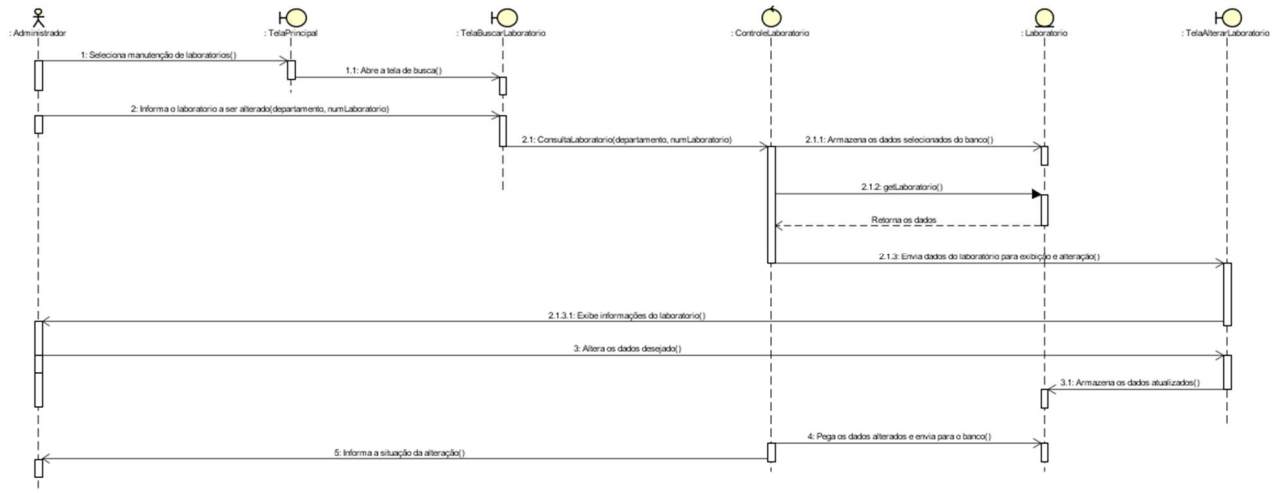


8.12 Buscar Header()



9. Diagrama de sequência

9.1 Alterar disponibilidade do laboratório



Atores: Administrador

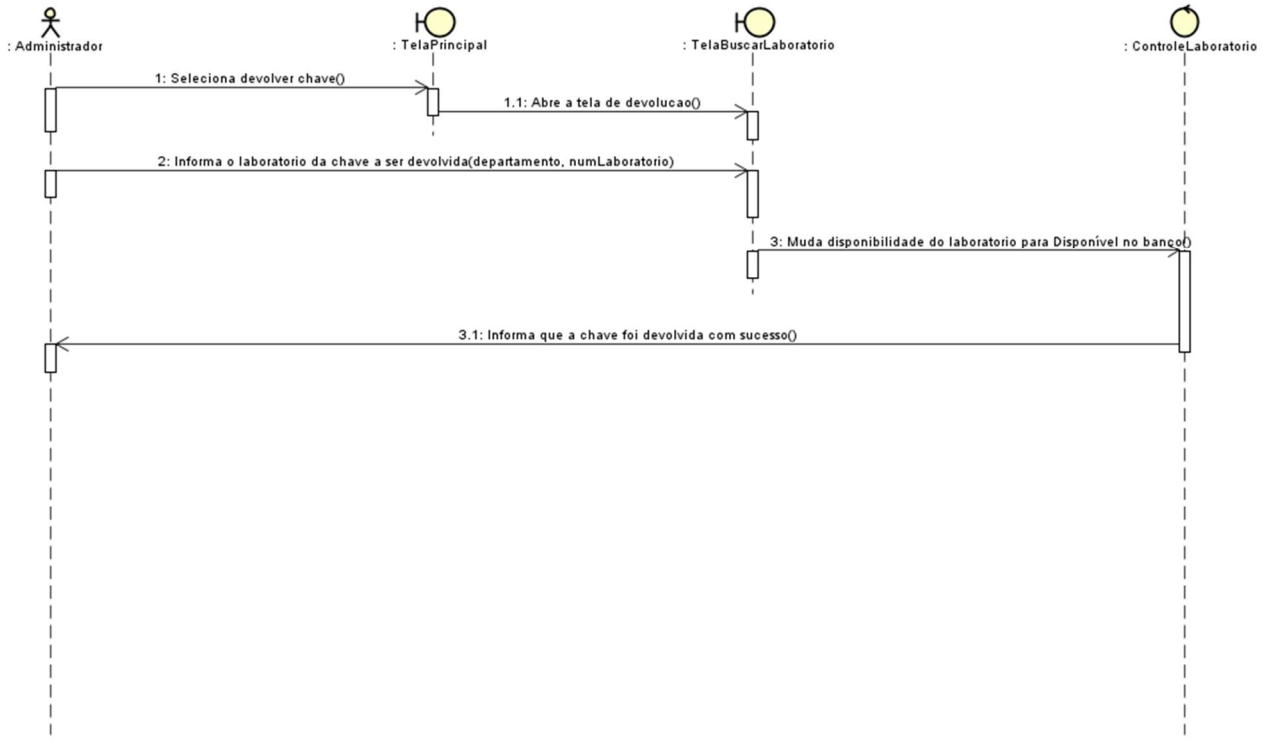
VIEWs: Tela Principal(Chavoso), Tela Escolher laboratório(Manutenção, Entrega de chave, Devolução de chave)

Controllers: ControleLaboratório

DAOs: Laboratório(DaoLaboratório)

OBS: O fluxo do diagrama referente ao caso de uso “alterar disponibilidade laboratório” tem como ator o administrador que irá interagir com a tela principal do programa, vale lembrar que essa interação pode ocorrer de três maneiras(manutenção, entregar chave, devolver chave), mas para diminuir a complexidade do diagrama foi utilizado uma generalização onde como em todos os casos o administrador teria que escolher qual laboratório teria manutenção, chave entregue ou chave devolvida. A partir disso foi representando apenas a generalização onde o autor irá interagir com a tela principal e a partir disso escolherá o laboratório para que este receba a alteração desejada.

9.2 Devolver Chave



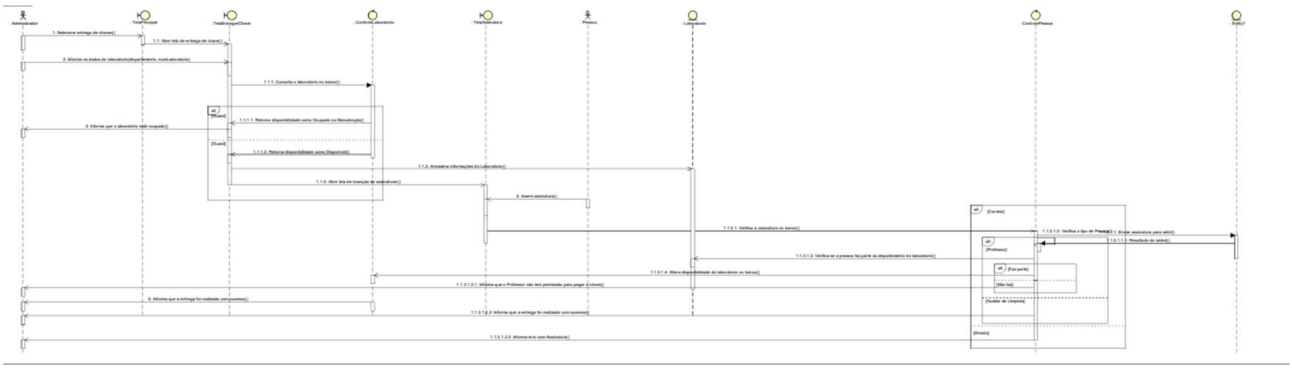
Atores: Administrador

VIEWs: Tela Principal(Chavoso), Tela Escolher laboratório(Devoluçao de chave)

Controllers: ControleLaboratório

DAOs: Para utilização do dao, o ControleLaboratório irá iniciar o fluxo de “alterar disponibilidade laboratório()” e esse usará conexão com o banco e retornará apenas a confirmação

9.4 Entregar chave



Atores: Administrador, Pessoa

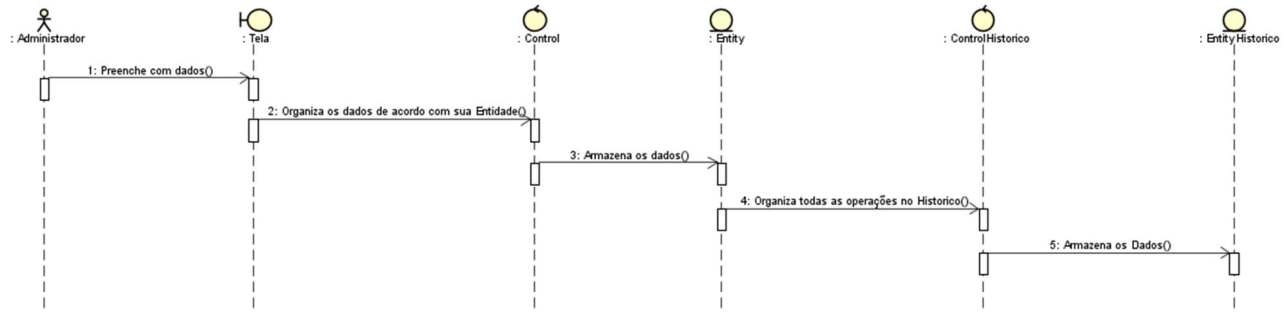
VIEWs: Tela Principal(Chavoso), Tela EntregarChave, Tela Assinatura

Controllers: ControleLaboratório, Controle Pessoa

DAOs: Laboratório, entity7(Pessoa)

OBS: O caso de uso “validar assinatura” é incluso dentro desse fluxo, ou seja, esse fluxo possui dois casos de uso.

9.5 Histórico



Atores: Administrador

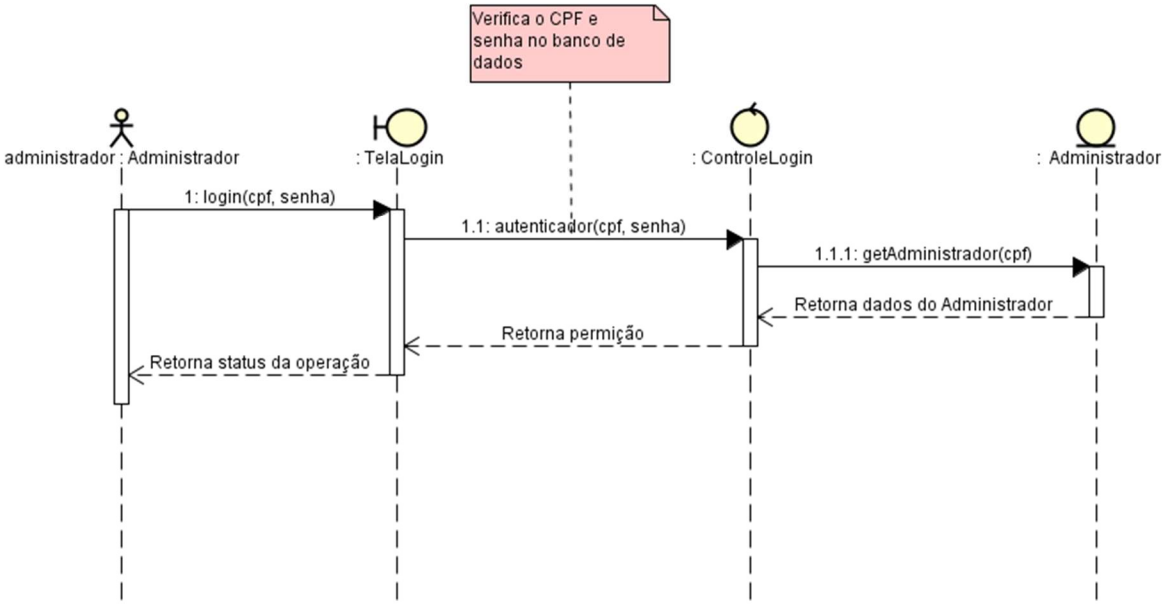
VIEWs: Tela(Chavoso)

Controllers: Control(Chavoso), ControlHistorico(Gerenciador_Historico)

DAOs: EntityHistorico, Entity(Chavoso)

OBS: Esse fluxo não fica visível para os usuários do sistema, mas ele necessita de um administrador para acontecer, onde conforme o administrador realiza ações no sistema ele vai guardando essas ações no banco de dados e o ControlHistorico() irá organizar essa ações para que seja possível buscar ações que ocorreram no sistema.

9.6 Login



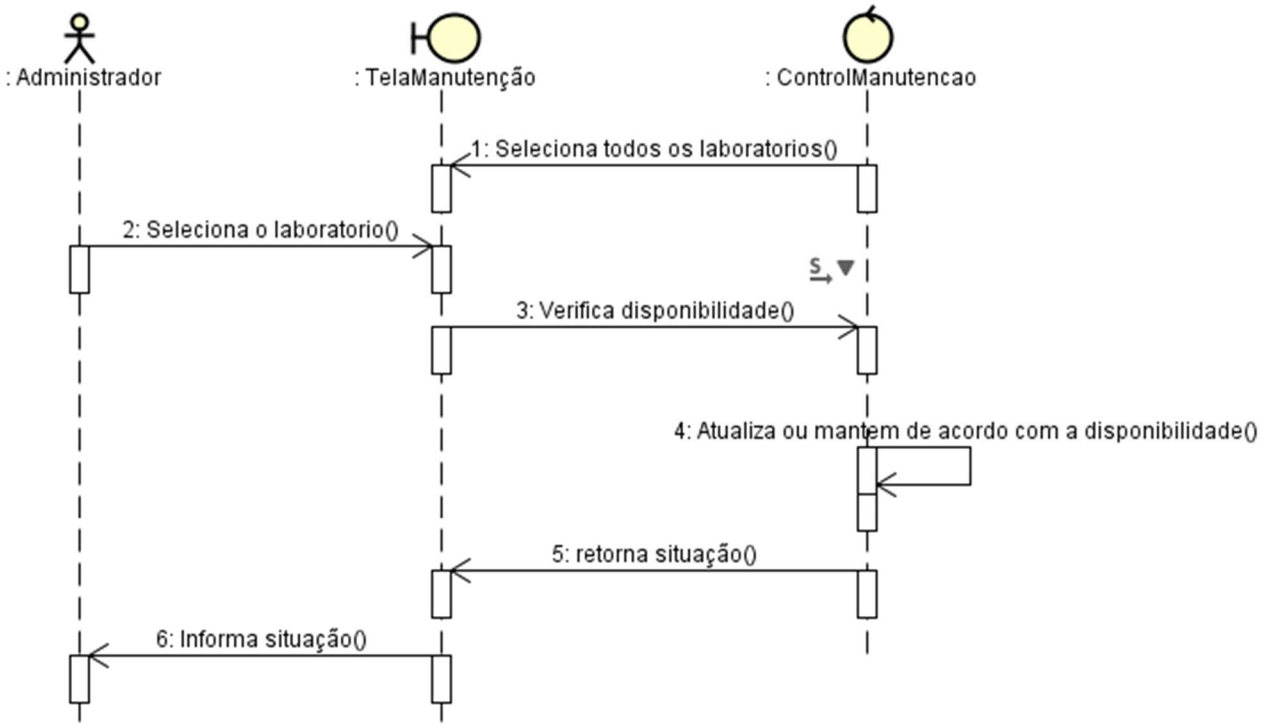
Atores: Administrador

VIEWs: TelaLogin

Controllers: ControlLogin

DAOs: Administrador

9.7 Manutenção



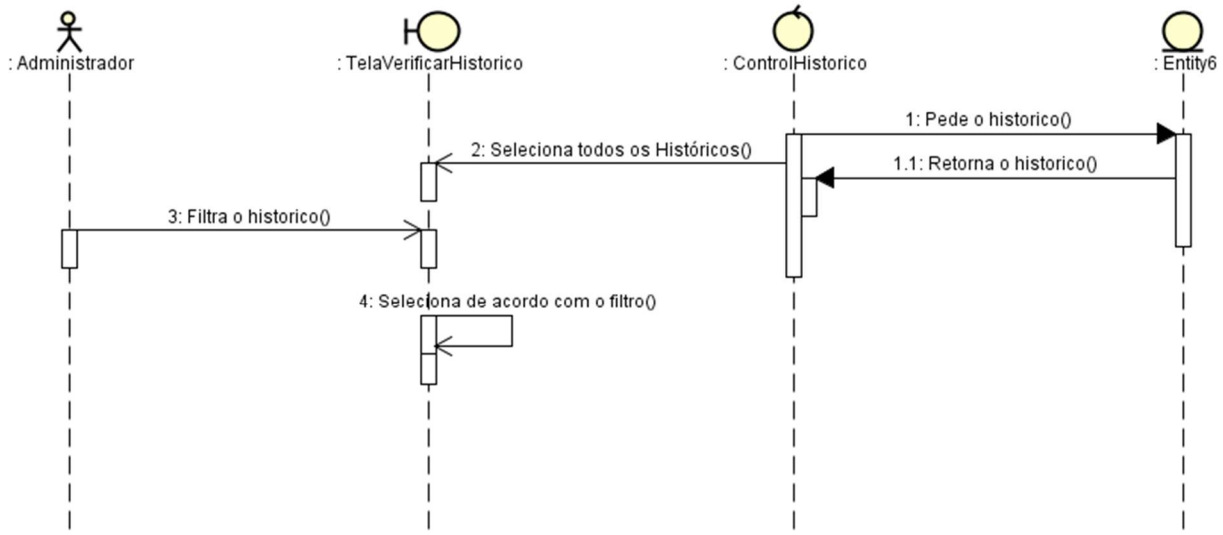
Atores: Administrador

VIEWs: TelaManutenção

Controllers: ControlManutencao

DAOs: Para utilização do dao, o ControlManutencao irá iniciar o fluxo de “alterar disponibilidade laboratório()” e esse usará conexão com o banco e retornará apenas a confirmação

9.8 Verifica histórico



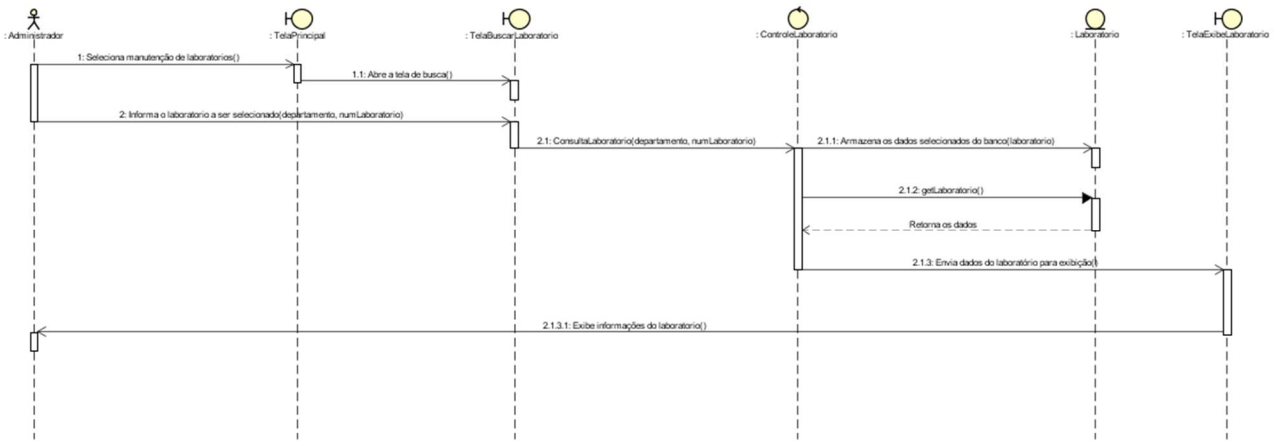
Atores: Administrador

VIEWs: TelaVerificarHistorico

Controllers: ControlHistorico

DAOs: Entity6(DaoHistorico)

9.9 Verifica disponibilidade histórico



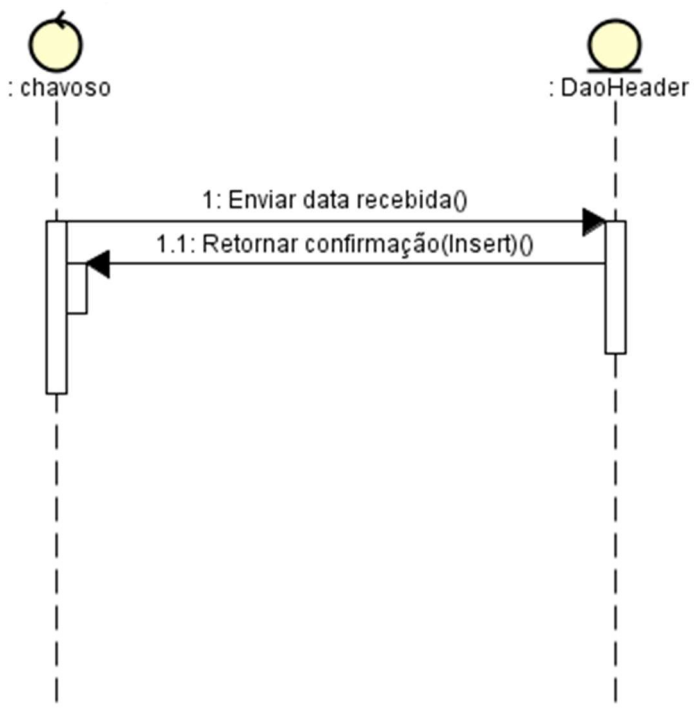
Atores: Administrador

VIEWs: TelaPrincipal,TelaBuscaLaboratorio, TelaExibiLaboratorio

Controllers: ControleLaboratorio

DAOs: Laboratório

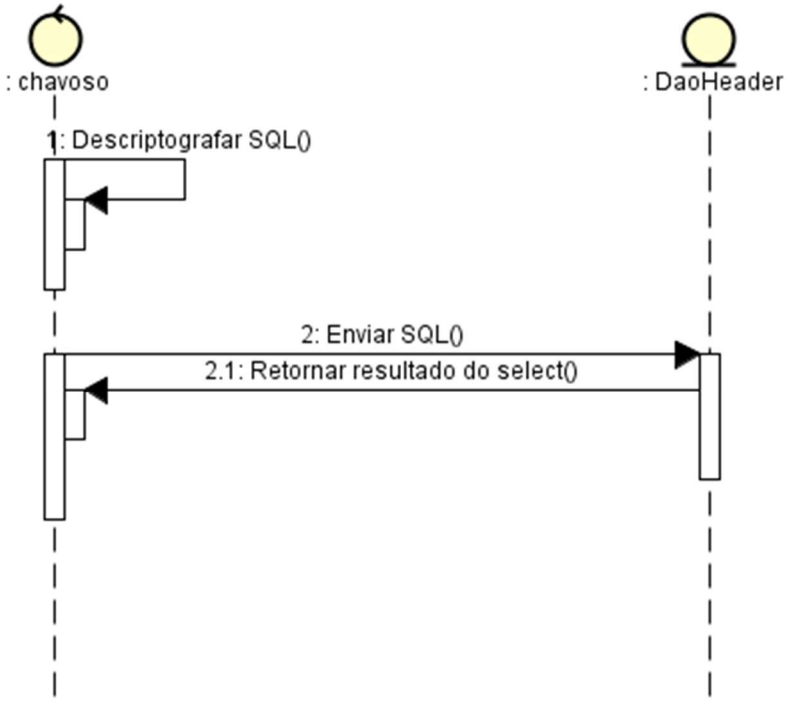
9.10 Gerar Header



DAOs: Header

Controllers: Chavoso

9.11 Buscar Header



DAOs: Header

Controllers: Chavoso