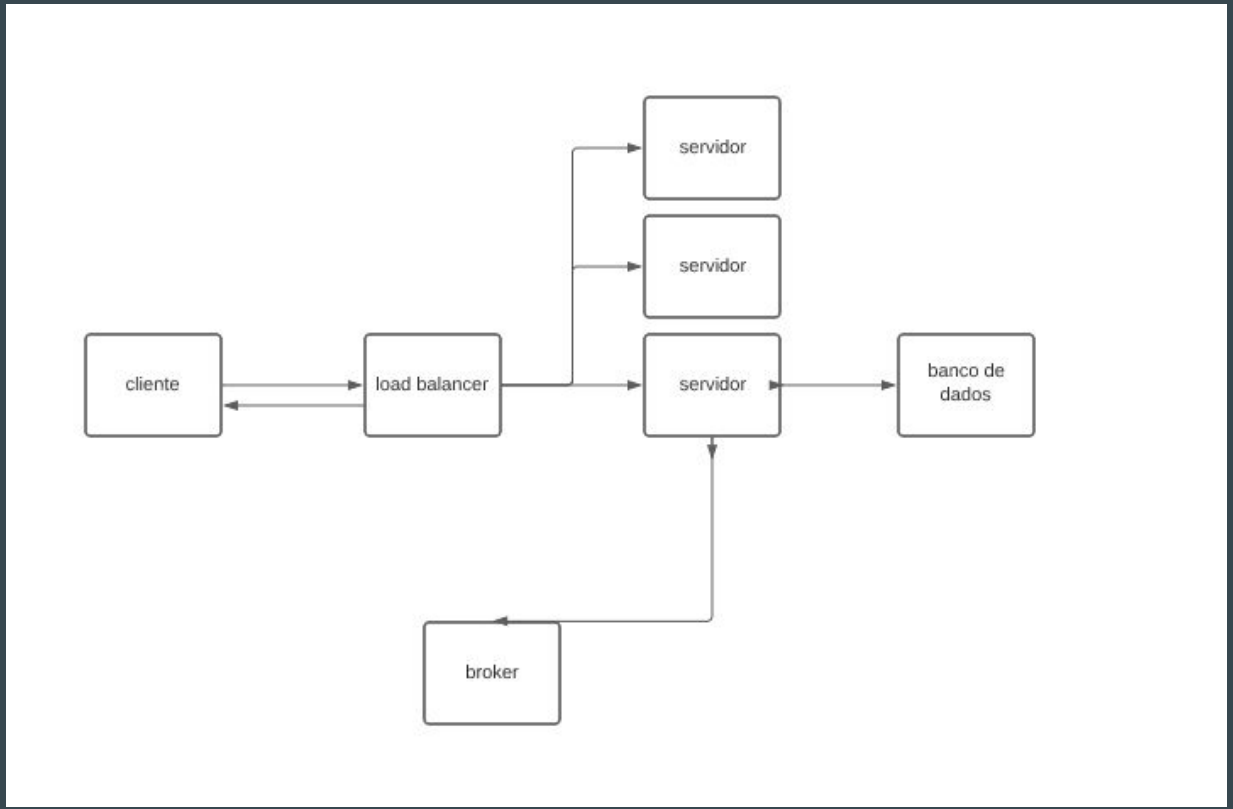
Projeto de Software

Esther de Mattos
Matheus Vieira
Pablo Almeida
Marcos Novaes
Lucas David
Leonardo Tramont

Minimundo

Após identificar as personas, identificamos o mini mundo a ser modelado, que posteriormente será utilizado para definição de requisitos funcionais do sistema.
Uma pessoa compra ou aluga uma casa para morar sozinha. Se depara com diversas questões inerentes à manutenção da mesma.
O software permite quem está morando sozinho gerenciar aspectos importantes da sua residência
Uma casa tem diversos cômodos, com diversas necessidades de manutenção e móveis
Uma pessoa que mora numa casa consome mantimentos e produtos
Uma pessoa que mora numa casa faz manutenção nos cômodos e nos móveis, e comprar mantimentos para seu sustento
A manutenção de uma residência implica em gastos financeiros
A pessoa que mora sozinha tem que controlar e gerenciar suas rendas e suas despesas.
A pessoa tem que ter um relatório de despesas para controle financeiro
A pessoa tem que conhecer os cômodos da casa e os aspectos de limpeza que cada um possui
A pessoa tem que saber as necessidades de manutenção de cada cômodo e deve saber quando realizá-los
A pessoa tem que saber o estoque de mantimentos e manter atualizado para não ficar sem produtos de limpeza e alimentação

ORGANIZAÇÃO DO SISTEMA



Arquitetura

Arquitetura em 3 camadas

Desta forma, conseguimos isolar a camada de lógica de negócio para servir aos diferentes clientes, como requisitado pelo requisito não funcional RNF1 e RNF2. Dessa forma, também não podemos usar a arquitetura MVC, pois a manutenção da capacidade de responder a diversas plataformas ia ser bastante complexa e propensa a se deteriorar

Para satisfazer o RNF3, a arquitetura em camadas também mostra alguns benefícios, visto que com a camada de banco de dados isolada, conseguimos melhorar a segurança desta sozinha e e tratar da segurança a nível de lógica na camada de lógica, deixando assim a camada cliente, que vai ser uma camada fina, menos propensa a vulnerabilidades

Pensando na fase de desenvolvimento do projeto, a arquitetura em camadas, com cada camada isolada, favorece a velocidade e produtividade do time por estarem trabalhando em projetos separados.

Arquitetura

Arquitetura Publish Subscribe

Esse padrão vai ser utilizado para resolver o RNF4, onde ele não pode fazer uma requisição para a camada de back-end pedindo o relatório e ficar esperando (as vezes por muito tempo) até que fique pronto. Desta forma, é gerado um evento que será consumido pelo servidor para gerar o relatório assincronamente, desacoplando assim o espaço e o tempo.

Em caso de erro na geração do relatório esse padrão também é interessante, pois o evento vai ficar sendo disparado após intervalos de tempo até que seja consumido pelo servidor, desta forma não “perdendo” a requisição do usuário em alguma eventual falha

Arquitetura

Arquitetura Orientada a mensagens

Esse padrão vai ser utilizado para resolver o RNF5, onde o usuário precisa consumir mensagens não lidas de uma fila de mensagens, mesmo que ele entre apos o evento de geração da notificação, então quando a mensagem for gerada, será inserida numa fila que deve ser consumida pelo cliente ao entrar no app novamente

Esse padrão garante resiliência em caso da requisição de busca de notificação, pois as mensagens vão estar persistidas na fila

Arquitetura

Para atender ao RNF7, foi adicionado uma camada de load balancer e redundancia de servidores de processamento de logica de dados, desta forma, mesmo que o volume de dados esteja cresça muito, o volume é distribuido ao longo dos servidores

Problema de consistencia nao foi avaliado como critico pois dois usuarios distintos vao sempre acessar registros diferentes de banco de dados, este que nao tem redundancia

Mecanismo de Persistência

O mecanismo de persistência escolhido é o PRISMA ORM, que é bastante conhecido na comunidade NODE.JS, e tem bastante recursos para trabalhar com bancos relacionais conhecidos como POSTGRESQL, MYSQL, entre outros. O mapeamento dos modelos e a conexão com o banco de dados serão implementados na camada de back-end onde fica a lógica e os modelos

Ele também se integra bem com o express, framework escolhido para fornecer as rotas onde os clientes podem acessar os controladores do back-end

Requisitos Funcionais

- RF1 USUÁRIO SE CADASTRA NA APLICAÇÃO POR MEIO DO CADASTRO
- RF2 USUÁRIO ACESSA A APLICAÇÃO POR MEIO DO LOGIN
- RF3 USUÁRIO ATUALIZA SEUS DADOS NA APLICAÇÃO
- RF4 USUÁRIO MANTÉM VALORES DE RENDA NA APLICAÇÃO
- RF5 USUÁRIO MANTÉM DESPESAS (CONTA) NA APLICAÇÃO
- RF6 SISTEMA CALCULA O NOVO VALOR DA RENDA E INFORMA AO USUÁRIO
- RF7 SISTEMA DEVERÁ ENVIAR ALERTA QUANDO A RENDA ESTIVER ZERADA
- RF8 SISTEMA DEVERÁ GERAR RELATÓRIO DE GASTOS FINANCEIROS E DE GASTOS DE RECURSOS (COMIDA, PRODUTOS), PROVISÃO FUTURA ENTRE OUTROS
- RF9 USUÁRIO MANTÉM CÔMODOS DA CASA NA APLICAÇÃO
- RF10 USUÁRIO MANTÉM O STATUS DE LIMPEZA DO CÔMODO NA APLICAÇÃO

Requisitos Funcionais

- RF11 O USUÁRIO PODERÁ CADASTRAR ALERTA SOBRE O CÔMODO, EXEMPLO: NECESSIDADE DE PINTURA, MANUTENÇÃO, ETC.
- RF12 SISTEMA DEVERÁ MANTER LISTA DE ALERTAS DOS CÔMODOS
- RF13 SISTEMA DEVE MANTER UMA LISTA DE MÓVEIS EM CADA CÔMODO
- RF14 USUÁRIO MANTÉM TIPOS DE PRODUTO NA APLICAÇÃO
- RF15 SISTEMA DEVE MANTER LISTA DE PRODUTOS CADASTRADOS
- RF16 USUÁRIO MANTÉM ESTOQUE, SELECIONANDO OS PRODUTOS CADASTRADOS E INFORMANDO QUANTIDADE
- RF17 SISTEMA DEVERÁ MANTER ATUALIZADO O ESTOQUE CRIADO
- RF18 O SISTEMA DEVERÁ ALERTAR ESTOQUE BAIXO QUANDO UM PRODUTO ESTIVER ABAIXO DE LIMITE INFERIOR CADASTRADO PELO USUÁRIO

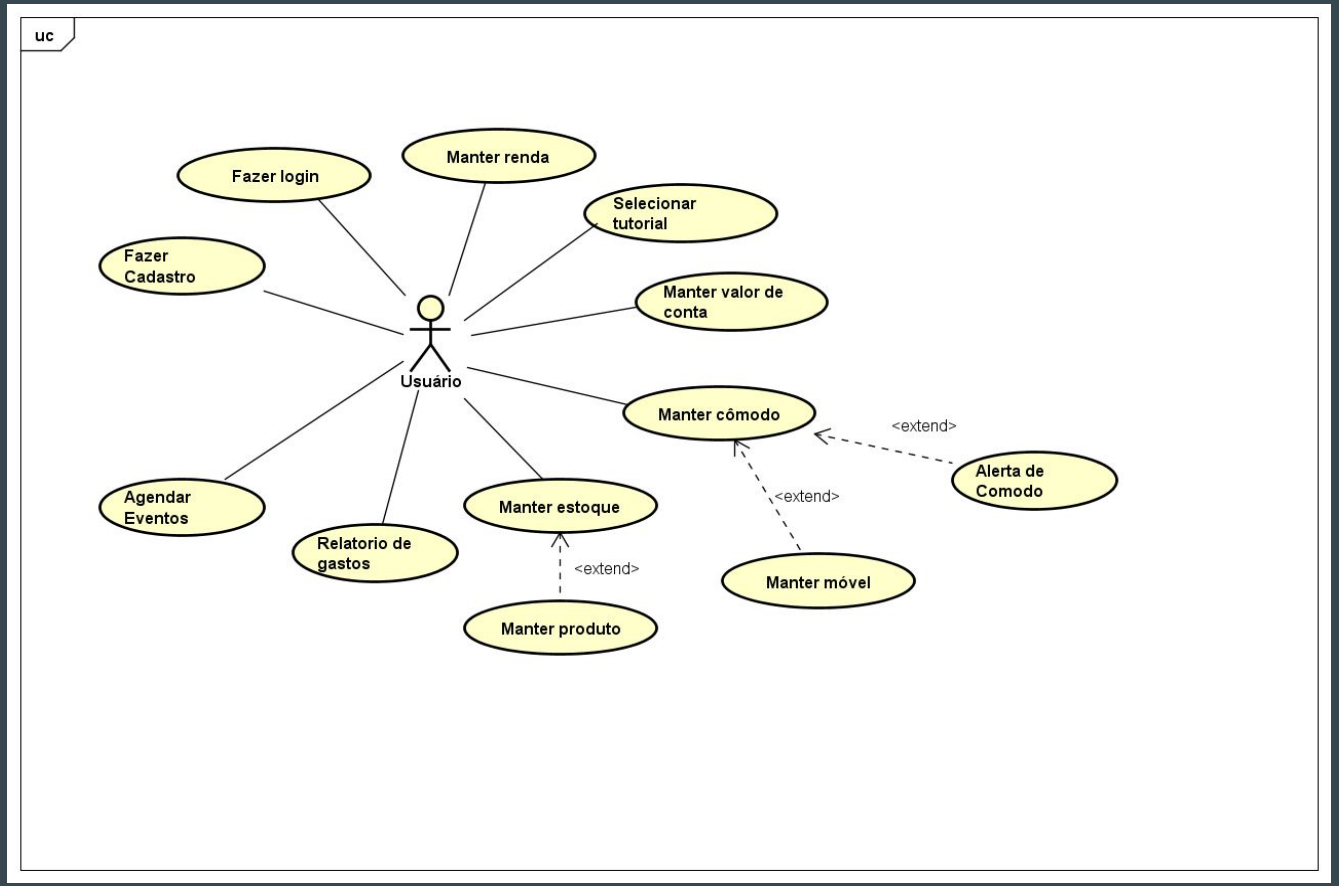
Requisitos Funcionais

- RF19 O SISTEMA DEVERÁ ALERTAR QUANDO O ESTOQUE DO PRODUTO ESTIVER ZERADO
- RF20 O USUÁRIO PODERÁ ACESSAR TUTORIAL PARA UMA ATIVIDADE COTIDIANA (EX. COMO COZINHAR UM PRATO ESPECÍFICO)
- RF21 O USUÁRIO PODERÁ AGENDAR EVENTOS EM UM CALENDÁRIO
- RF22 O SISTEMA DEVE EMITIR UMA NOTIFICAÇÃO NA CHEGADA DA DATA DE UM EVENTO DEFINIDO NO CALENDÁRIO

Requisitos Não Funcionais

- RNF1 A APLICAÇÃO DEVERÁ FUNCIONAR EM PLATAFORMA MOBILE
- RNF2 A APLICAÇÃO DEVERÁ SER ACESSADA VIA BROWSER HTTP/HTML
- RNF3 A APLICAÇÃO NÃO PODERÁ FORNECER AS INFORMAÇÕES DE USUÁRIO PARA AMBIENTES EXTERNOS
- RNF4 O USUÁRIO NÃO PODE ESPERAR SINCRONAMENTE PELO FIM DA GERAÇÃO DE RELATÓRIOS
- RNF5 O SISTEMA NÃO PODE PERDER NOTIFICAÇÕES CASO O CLIENTE ESTEJA FORA DO AR (OFFLINE) NO MOMENTO DA GERAÇÃO
- RNF6 QUALQUER USUARIO (NÃO TREINADO OU TREINADO) DEVE SER CAPAZ DE CADASTRAR UM CÔMODO EM MENOS DE 10 MINUTOS
- RNF7 O SISTEMA NAO PODE TER UMA LATENCIA MAIOR QUE 10% NOS PERIODOS DE MAIOR VOLUME DE ACESSO QUE A LATENCIA NO VOLUME MEDIO

Diagramas de Caso de Uso



Descrição de Caso de Uso - Fazer Login

Nome: Fazer Login

Objetivo: Autenticar o usuário ao acessar o sistema.

Atores: Usuários.

Pré-condições: O usuário deve estar cadastrado no sistema.

Fluxo principal:

1. O usuário informa as suas informações de login.
2. O sistema verifica a existência do cliente.[x]
3. O sistema redireciona o usuário para a página principal.

Fluxo alternativo:

[x] Uma falha acontece no processamento da solicitação

1. O sistema informa que ocorreu uma falha no processamento da solicitação.
2. Retorne ao passo 1 do Fluxo Principal

Pós-condições: O usuário acessa o sistema de forma integral.

Descrição de Caso de Uso - Fazer Cadastro

Nome: Fazer Cadastro

Objetivo: Cadastrar as informações de um novo usuário para acessar o sistema.

Atores: Usuários.

Pré-condições: O usuário deve acessar o sistema.

Fluxo principal:

1. O usuário clica na opção “fazer cadastro”.
2. O usuário insere as informações de cadastro e confirma. [x]
3. O sistema redireciona o usuário para a página principal já logado.

Fluxo alternativo:

[x] Uma falha acontece no processamento da solicitação

1. O sistema informa que ocorreu uma falha no processamento da solicitação.
2. Retorne ao passo 1 do Fluxo Principal

Pós-condições: O usuário acessa o sistema de forma integral.

Descrição de Caso de Uso - Manter Renda

Nome: Manter Renda

Objetivo: Realizar cadastro, alteração, ou remoção da renda que o usuário possui ;

Atores: Usuários.

Pré-condições: O usuário deve possuir acesso ao sistema.

Fluxo principal:

1. O usuário clica na opção de renda
2. O sistema exibe os dados já cadastrados e opções de cadastrar, alterar ou remover um valor de renda
3. O usuário seleciona a opção de cadastrar
4. O usuário insere o valor de renda
5. O usuário confirma o cadastro da informação
6. O sistema registra a informação e exibe uma confirmação [X]

Fluxo alternativo:

- **3.a - Atualizar renda:**

    3.1 O usuário seleciona a opção de alterar

    3.2 O usuário altera os dados da renda desejada dentre aquelas cadastradas

    3.3 O usuário confirma a alteração

    3.4 Retoma o passo 6 do fluxo principal

- **Remover renda:**

    3.1 O usuário seleciona a opção de remover

    3.2 O usuário remove o registro de renda desejada dentre aqueles cadastrados no passo 3 do fluxo principal

    3.3 O usuário confirma a remoção

    3.4 Retoma o passo 6 do fluxo principal

[x] Uma falha acontece no processamento da solicitação.

1. O sistema informa que ocorreu uma falha no processamento da solicitação.
2. Retorne ao passo 2 do Fluxo Principal.

Descrição de Caso - Manter Valor de Conta

Nome: Manter Valor de Conta.

Objetivo: Realizar cadastro, alteração, ou remoção das despesas que o usuário possui;

Atores: Usuários.

Pré-condições: O usuário deve possuir acesso ao sistema.

Fluxo principal:

1. O usuário clica na opção de Valor de Conta
2. O sistema exibe os dados já cadastrados e opções de cadastrar, alterar ou remover um valor de conta
3. O usuário seleciona a opção de cadastrar
4. O usuário insere o valor de conta
5. O usuário confirma o cadastro da informação
6. O sistema registra a informação e exibe uma confirmação [X]

Fluxo alternativo:

- **Atualizar conta:**

    3.1 O usuário seleciona a opção de alterar

    3.2 O usuário altera um dos valores de conta dentre aqueles cadastrados

    3.3 O usuário confirma a alteração

    3.4 Retoma o passo 6 do fluxo principal

- **Remover conta:**

    3.1 O usuário seleciona a opção de remover

    3.2 O usuário remove um dos valores de conta dentre aqueles cadastrados

    3.3 O usuário confirma a remoção

    3.4 Retoma o passo 6 do fluxo principal

[x] Uma falha acontece no processamento da solicitação.

1. O sistema informa que ocorreu uma falha no processamento da solicitação.
2. Retorne ao passo X do Fluxo Principal.

Pós-condições: As despesas do usuário são mantidas no sistema com sucesso.

Descrição de Caso - Manter Cômodo

Nome: Manter Cômodo.

Objetivo: Realizar cadastro, alteração, ou remoção de cômodos no sistema ;

Atores: Usuários.

Pré-condições: O usuário deve possuir acesso ao sistema.

Fluxo principal:

1. O usuário clica na opção de Cômodos.
2. O sistema exibe os dados já cadastrados (se já tiverem) e opções de cadastrar, alterar ou remover um cômodo
3. O usuário seleciona a opção de cadastrar
4. O usuário as informações do cômodo que vai inserir
5. O usuário confirma o cadastro da informação
6. O sistema registra a informação e exibe uma confirmação [X]

Fluxo alternativo:

- **3.a - Atualizar Cômodo:**

    3.1 O usuário seleciona a opção de alterar

    3.2 O usuário altera os dados do cômodo desejado dentre aqueles cadastrados.

    3.3 O usuário confirma a alteração

    3.4 Retoma o passo 6 do fluxo principal

- **Remover Cômodo:**

    3.1 O usuário seleciona a opção de remover

    3.2 O usuário remove o registro do cômodo desejado dentre aqueles cadastrados

    3.3 O usuário confirma a remoção

    3.4 Retoma o passo 6 do fluxo principal

[x] Uma falha acontece no processamento da solicitação.

1. O sistema informa que ocorreu uma falha no processamento da solicitação.

Retorne ao passo 2 do Fluxo Principal.

Pós-condições: O cômodo do usuário é mantido no sistema com sucesso.

Descrição de Caso - Manter Móvel

Nome: Manter Valor de Conta.

Objetivo: Realizar cadastro, alteração, ou remoção de móveis que o cômodo possui ;

Atores: Usuários.

Pré-condições: O usuário deve possuir acesso ao sistema.

Fluxo principal:

1. O usuário clica na opção de *Móveis* em um dos *Cômodos* cadastrados
2. O sistema exibe os dados já cadastrados (se já tiverem) e opções de cadastrar, alterar ou remover um cômodo
3. O usuário seleciona a opção de cadastrar
4. O usuário informa as informações do móvel que vai inserir
5. O usuário confirma o cadastro da informação
6. O sistema registra a informação e exibe uma confirmação [X]

Fluxo alternativo:

- **3.a - Atualizar Móvel:**

    3.1 O usuário seleciona a opção de alterar

    3.2 O usuário altera os dados do móvel desejado dentre aqueles cadastrados.

    3.3 O usuário confirma a alteração

    3.4 Retoma o passo 6 do fluxo principal

- **Remover Móvel:**

    3.1 O usuário seleciona a opção de remover móvel

    3.2 O usuário remove o registro do movel desejado dentre aqueles cadastrados

    3.3 O usuário confirma a remoção

    3.4 Retoma o passo 6 do fluxo principal

[x] Uma falha acontece no processamento da solicitação.

1. O sistema informa que ocorreu uma falha no processamento da solicitação.

Retorne ao passo 2 do Fluxo Principal.

Pós-condições: Os Móveis do usuário são mantidos no sistema com sucesso.

Descrição de Caso - Manter Produto

Nome: Manter Estoque.

Objetivo: Realizar cadastro, alteração, ou remoção dos estoques de produtos no sistema ;

Atores: Usuários.

Pré-condições: O usuário deve possuir acesso ao sistema.

Fluxo principal:

1. O usuário clica na opção de produto
2. O sistema exibe os produtos já cadastrados e opções de cadastrar, alterar ou remover um produto
3. O usuário seleciona a opção de cadastrar
4. O usuário insere o produto desejado
5. O usuário confirma o cadastro do produto
6. O sistema registra a informação e exibe uma confirmação [X]

Fluxo alternativo:

- **Atualizar estoque:**
    - 3.1 O usuário seleciona a opção de alterar
    - 3.2 O usuário altera os dados de um produto desejado dentre aqueles cadastrados
    - 3.3 O usuário confirma a alteração
    - 3.4 Retoma o passo 6 do fluxo principal

- **Remover estoque:**
    - 3.1 O usuário seleciona a opção de remover
    - 3.2 O usuário remove um produto dentre aqueles cadastrados
    - 3.3 O usuário confirma a remoção
    - 3.4 Retoma o passo 6 do fluxo principal

[x] Uma falha acontece no processamento da solicitação.

1. O sistema informa que ocorreu uma falha no processamento da solicitação.
2. Retorne ao passo X do Fluxo Principal.

Pós-condições: Os produtos do usuário são mantidos no sistema com sucesso.

Descrição de Caso - Manter Produto no Estoque

Nome: Manter Produto no Estoque.

Objetivo: Realizar cadastro e remoção dos produtos em estoque no sistema ;

Atores: Usuários.

Pré-condições: O usuário deve possuir acesso ao sistema.

Fluxo principal:

1. O usuário clica na opção de estoque
2. O sistema exibe os dados já cadastrados e opções de cadastrar, alterar ou remover um item
3. O usuário seleciona a opção de cadastrar
4. O usuário insere o produto no estoque
5. O usuário confirma o cadastro do produto no estoque
6. O sistema registra a informação e exibe uma confirmação [X]

Fluxo alternativo:

- **Remover produto no estoque:**
    - 3.1 O usuário seleciona a opção de remover
    - 3.2 O usuário remove um produto dentre aqueles cadastrados do estoque
    - 3.3 O usuário confirma a remoção
    - 3.4 Retoma o passo 6 do fluxo principal

[x] Uma falha acontece no processamento da solicitação.

1. O sistema informa que ocorreu uma falha no processamento da solicitação.
2. Retorne ao passo X do Fluxo Principal.

Pós-condições: O produto é mantido no estoque do usuário com sucesso.

Descrição de Caso - Selecionar Tutorial

Nome: Selecionar Tutorial.

Objetivo: Permitir que o usuário acesse tutoriais.

Atores: Usuários.

Pré-condições: O usuário deve possuir acesso ao sistema.

Fluxo principal:

1. Usuário seleciona a página “Tutoriais”.
2. O usuário busca o tutorial desejado dentre os disponíveis no sistema.
3. O usuário seleciona o tutorial desejado.
4. O sistema mostra a página com o tutorial selecionado.

Fluxo alternativo:

[x] Uma falha acontece no processamento da solicitação.

1. O sistema informa que ocorreu uma falha no processamento da solicitação.
2. Retorne ao passo 1 do Fluxo Principal.

Pós-condições: O usuário tem acesso ao conteúdo da página com o tutorial selecionado.

Descrição de Caso - Agendar evento

Nome: Agendar evento.

Objetivo: Permitir que o usuário agende um evento em seu calendário.

Atores: Usuários.

Pré-condições: O usuário deve possuir acesso ao sistema.

Fluxo principal:

1. O usuário seleciona o calendário na página principal.
2. O usuário seleciona uma data dentro do calendário.
3. O usuário cadastra um evento na data selecionada.
4. O usuário informa a descrição do evento e confirma o cadastro.

Fluxo alternativo:

[x] Uma falha acontece no processamento da solicitação.

1. O sistema informa que ocorreu uma falha no processamento da solicitação.
2. Retorne ao passo 1 do Fluxo Principal.

Pós-condições: O usuário cadastra um evento em seu calendário com sucesso.

Diagrama de Classe

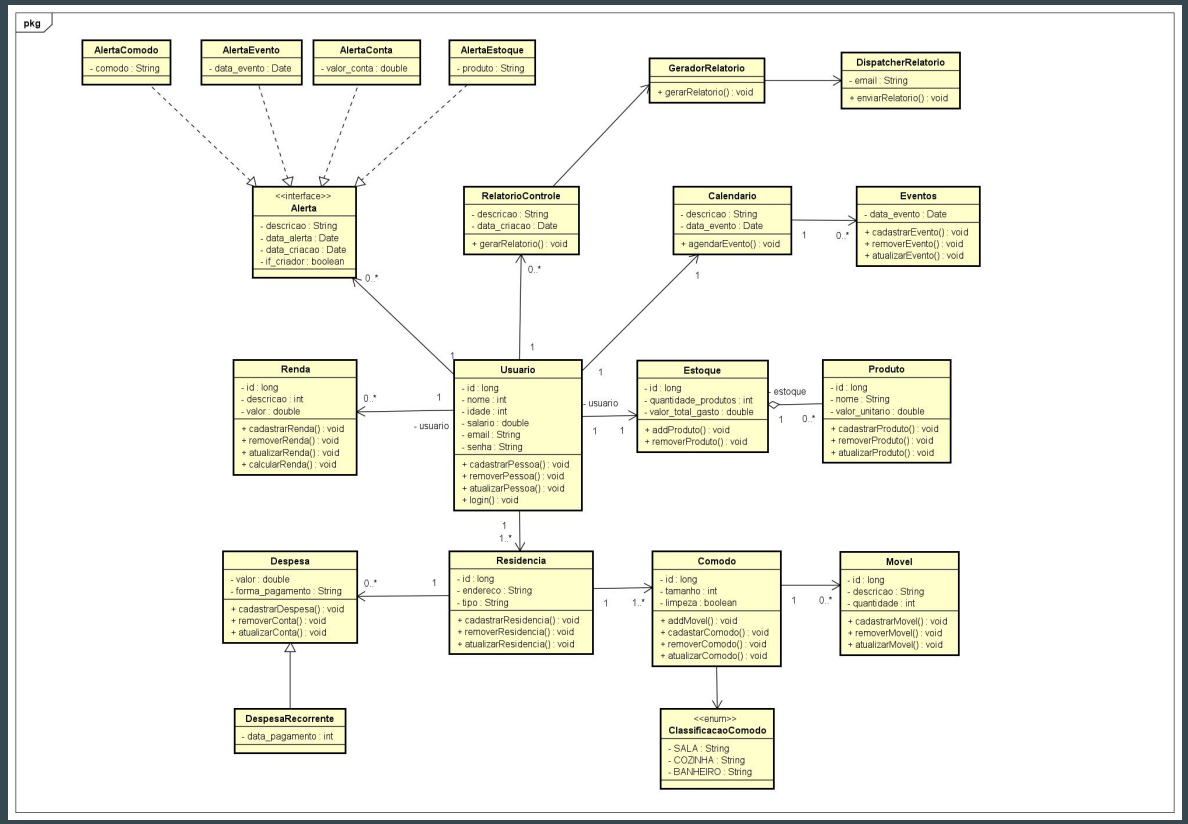
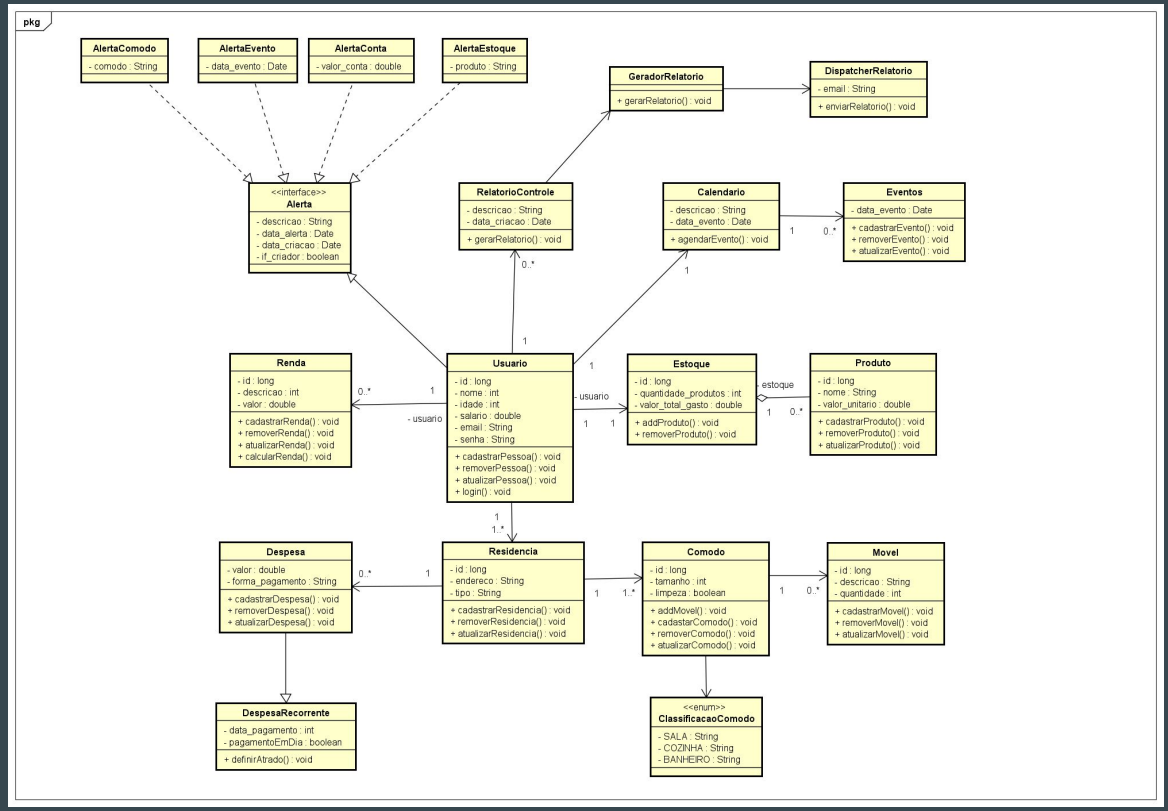
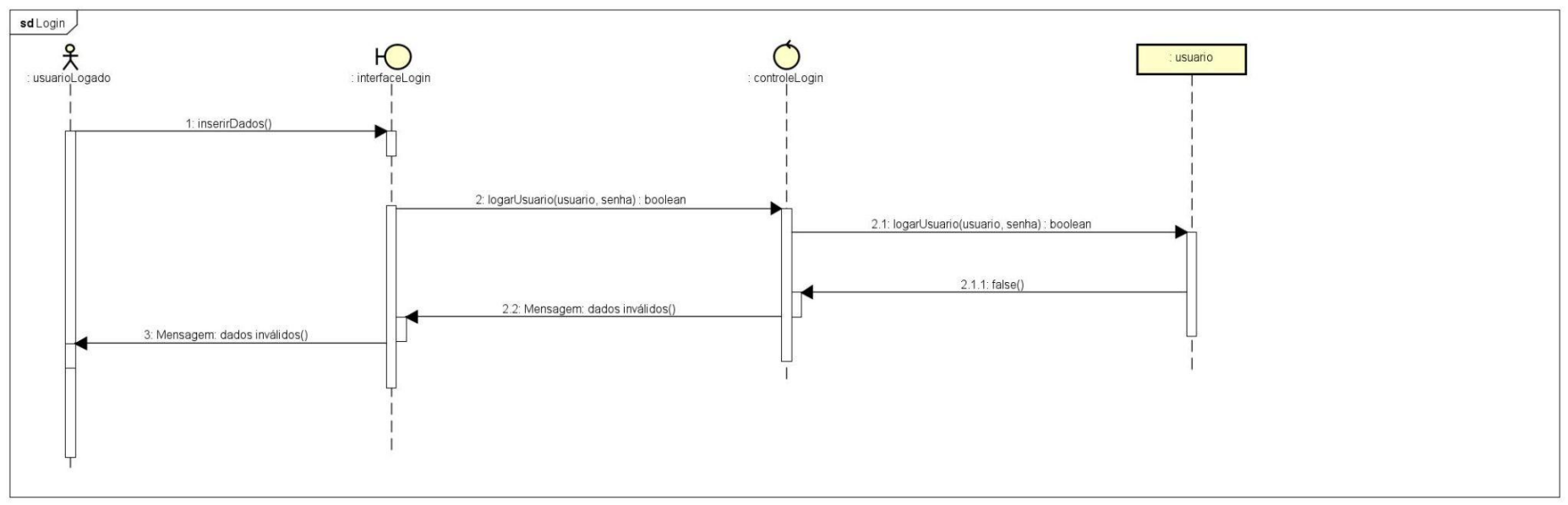


Diagrama de Classe(novo)

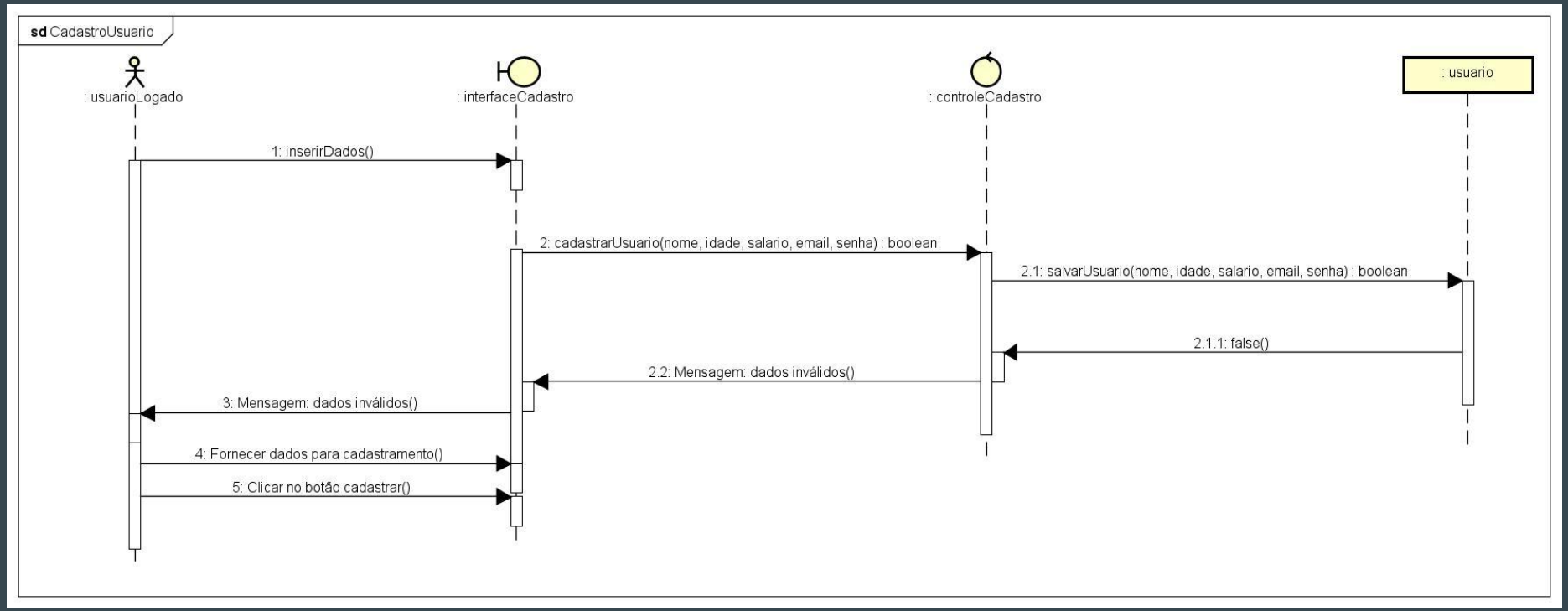


Diagramas de Sequência

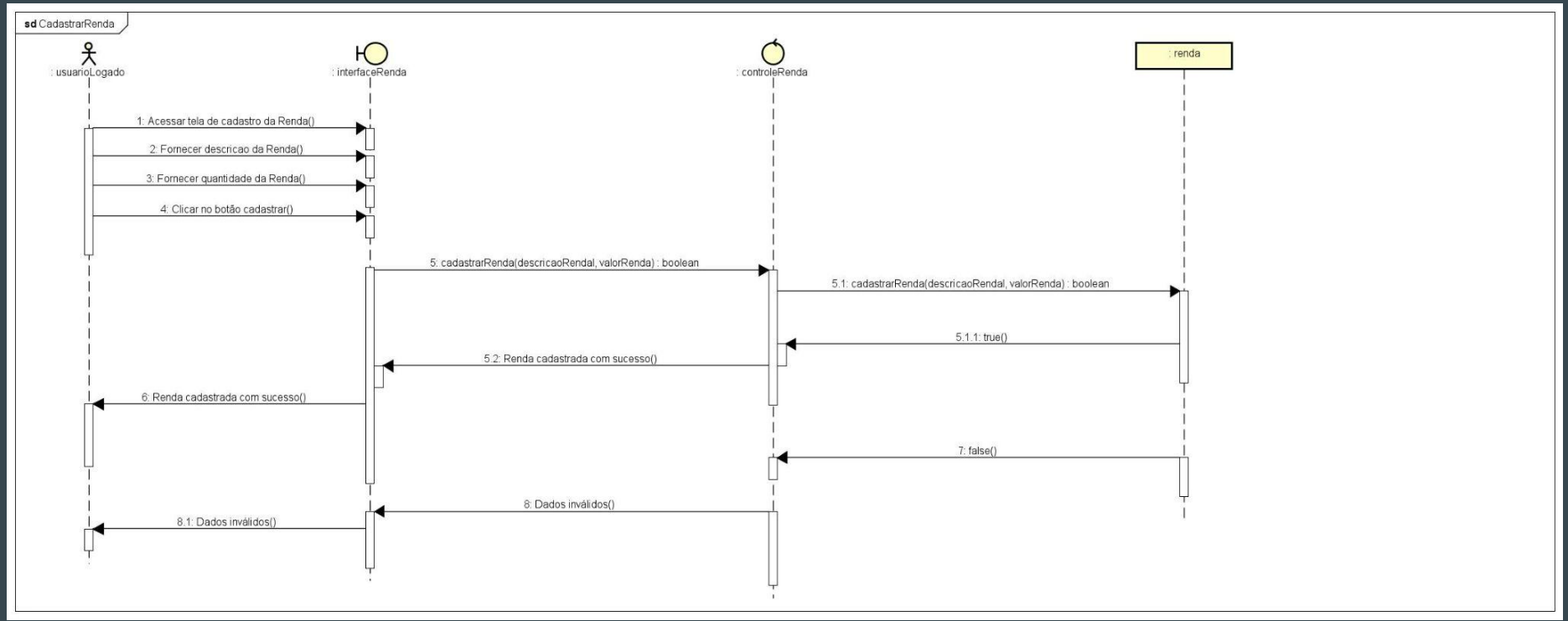
Login


sd Login
: usuarioLogado
: interfaceLogin
: controleLogin
: usuario
1: inserirDados()
2: logarUsuario(usuario, senha) : boolean
2.1: logarUsuario(usuario, senha) : boolean
2.1.1: false()
2.2: Mensagem: dados inválidos()
3: Mensagem: dados inválidos()

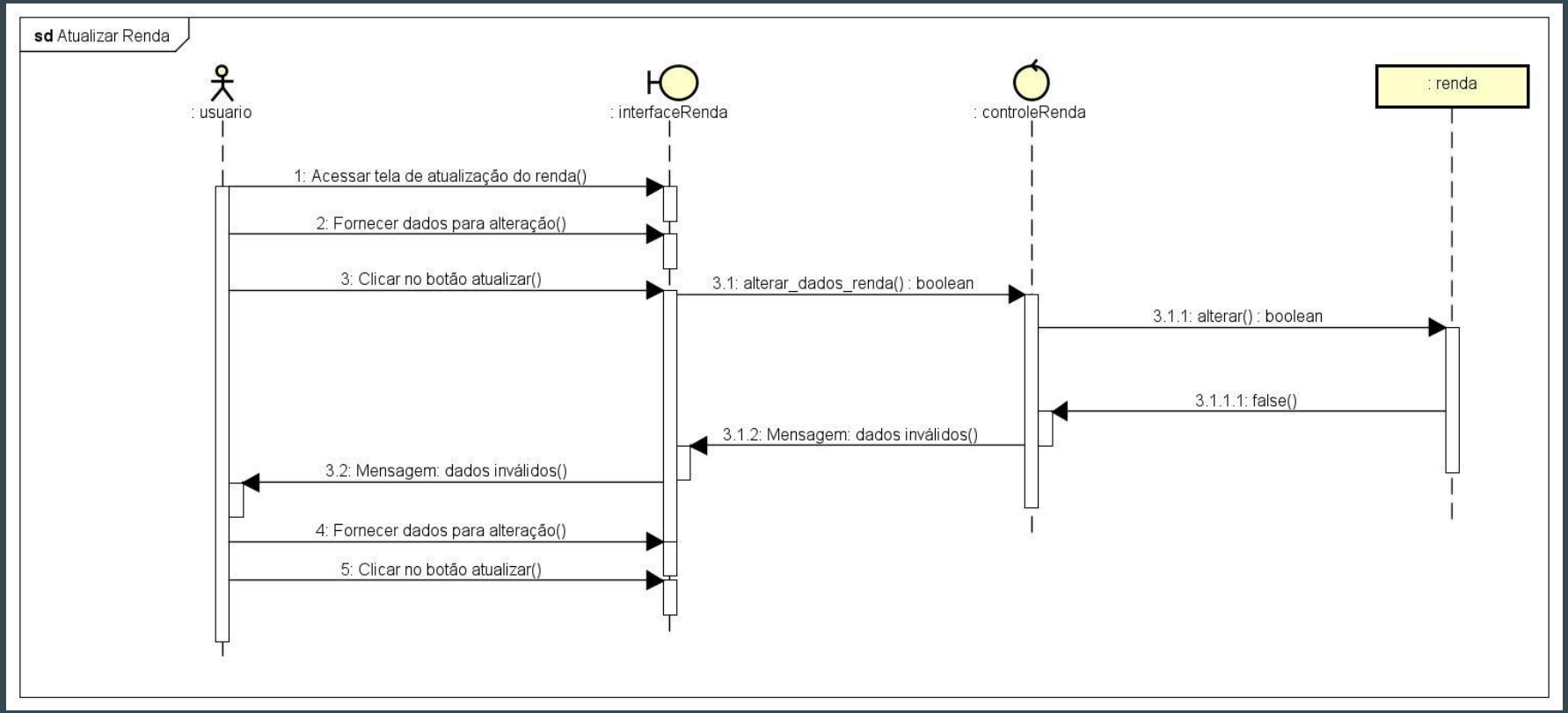
Cadastro de Usuário



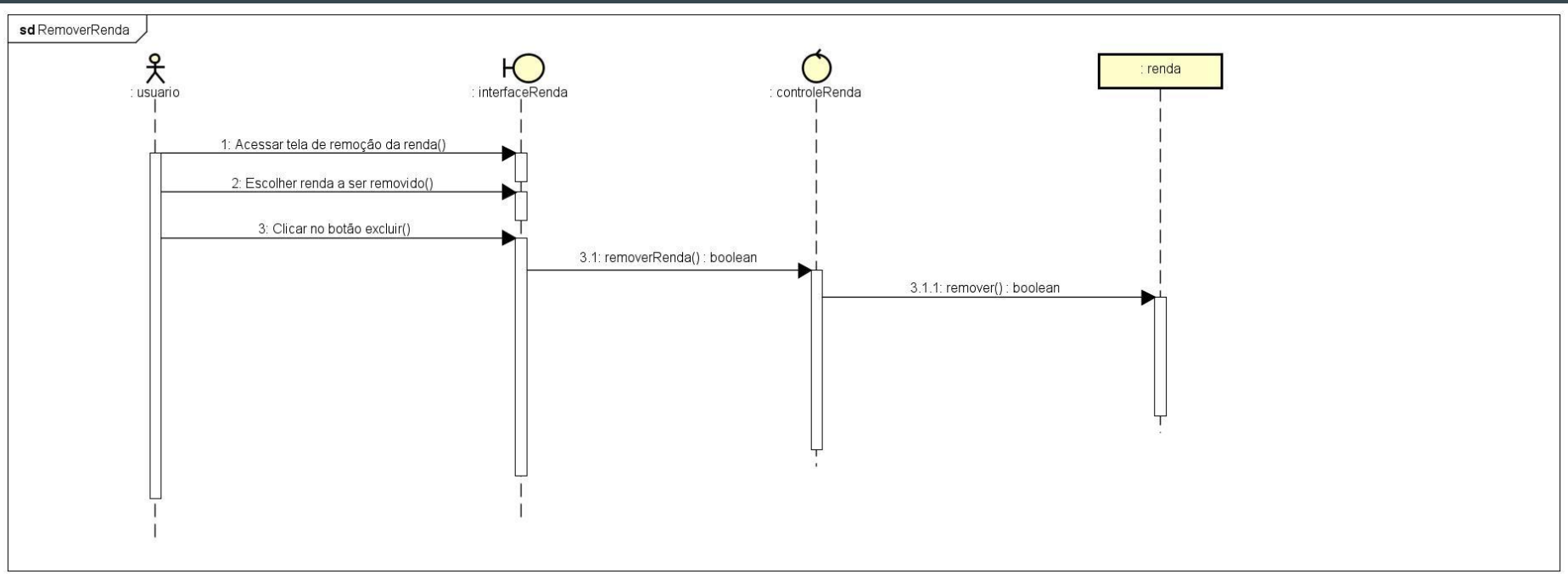
Manter Renda - Cadastro



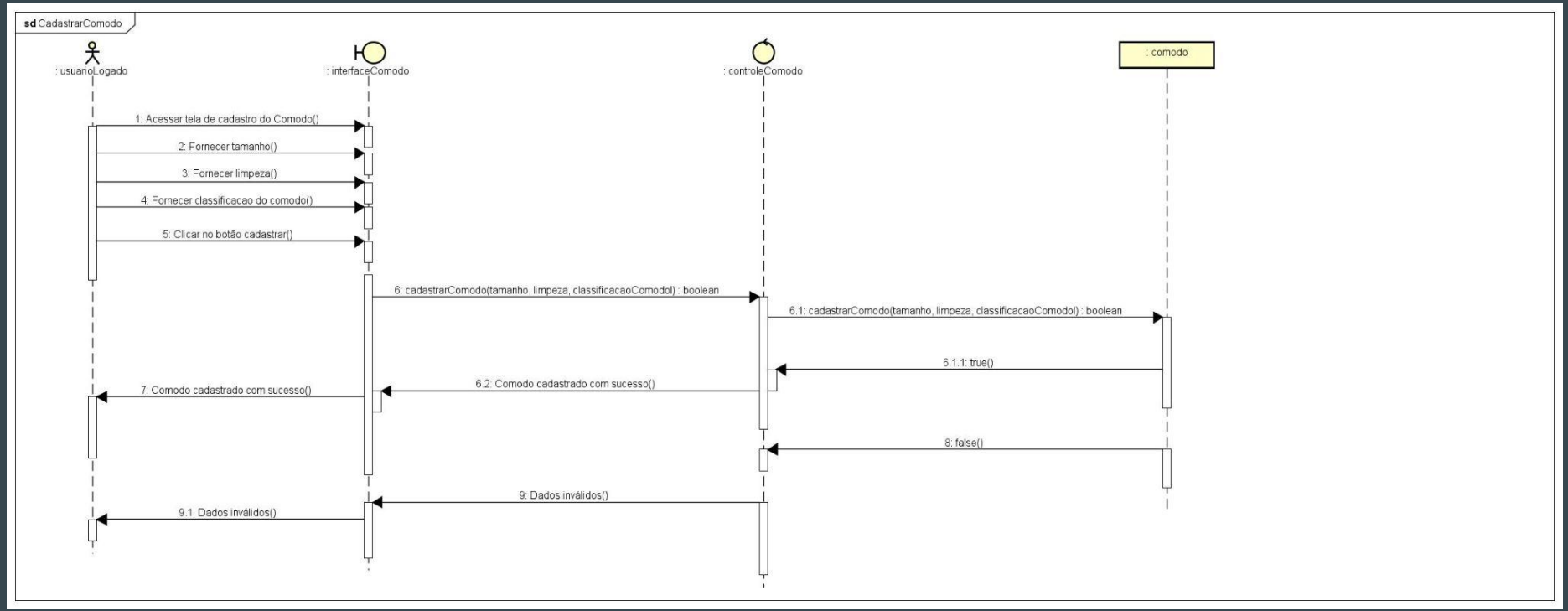
Manter Renda - Atualizar



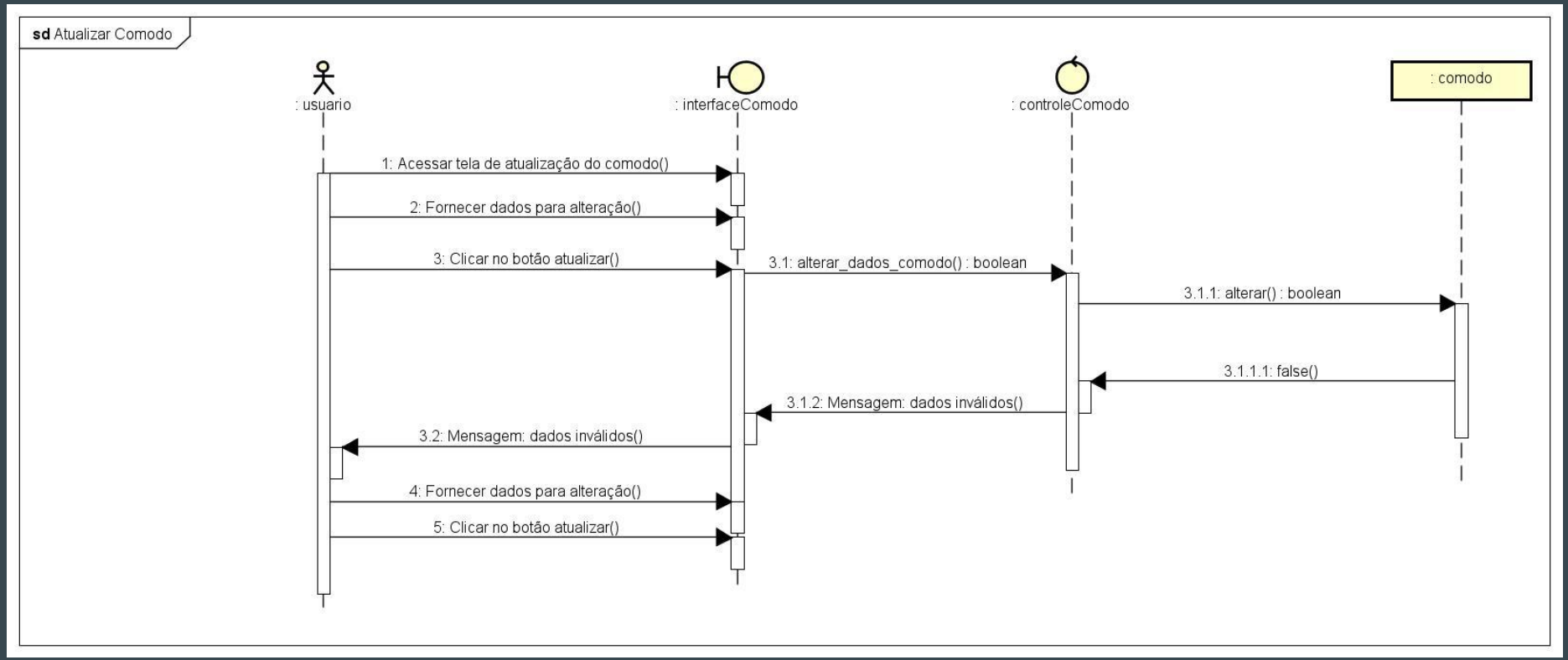
Manter Renda - Remover



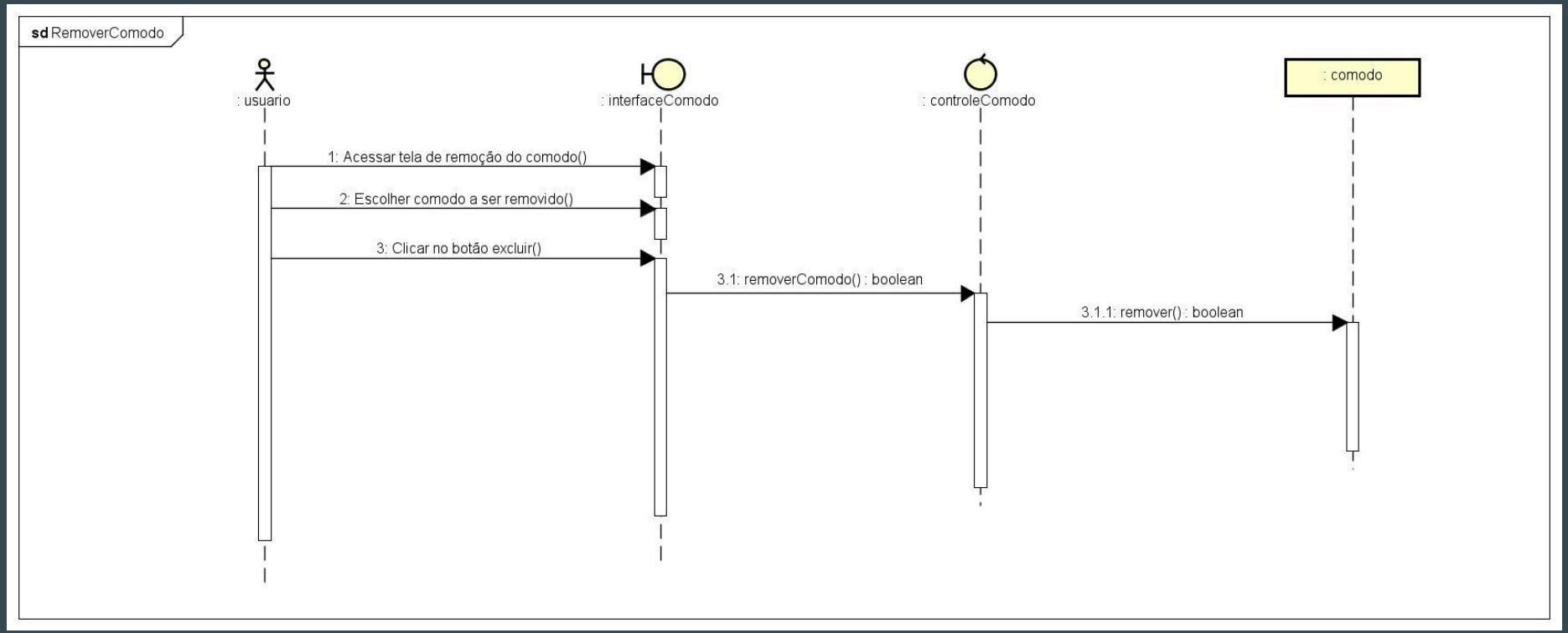
Manter Cômodo - Cadastro



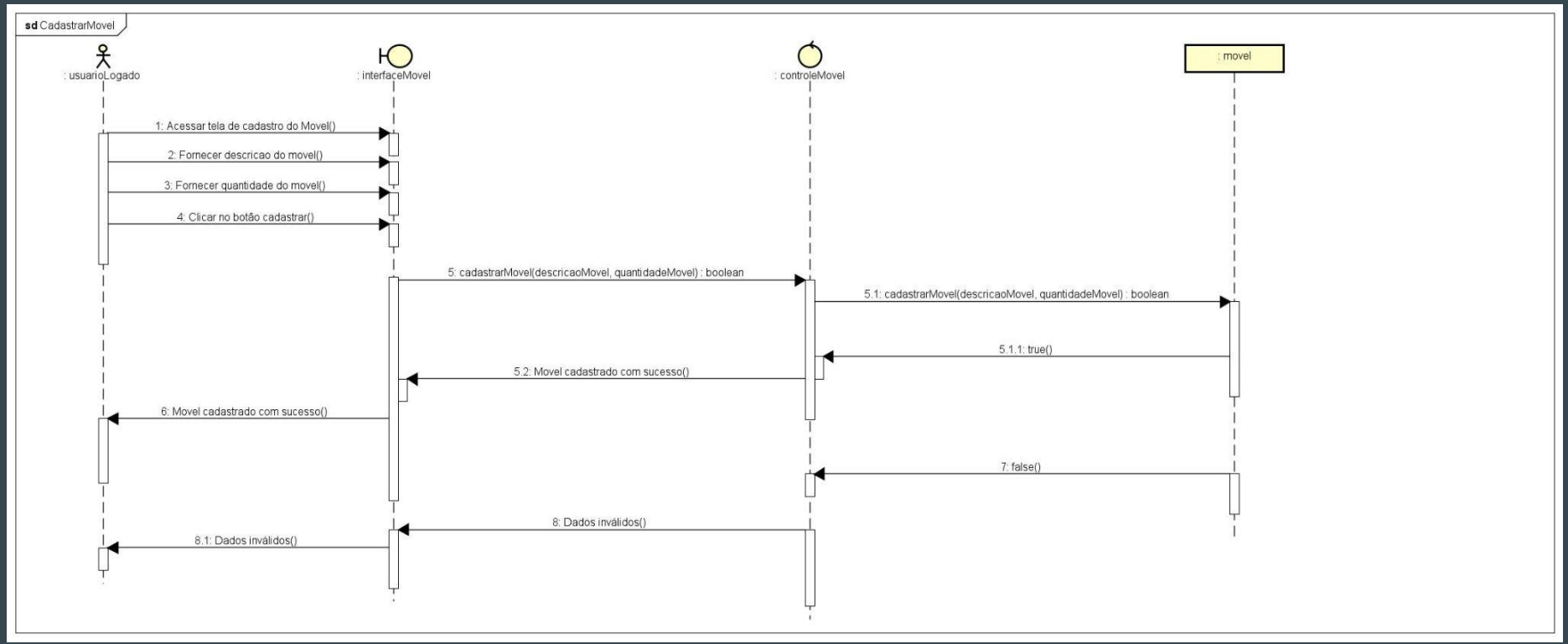
Manter Cômodo - Atualizar



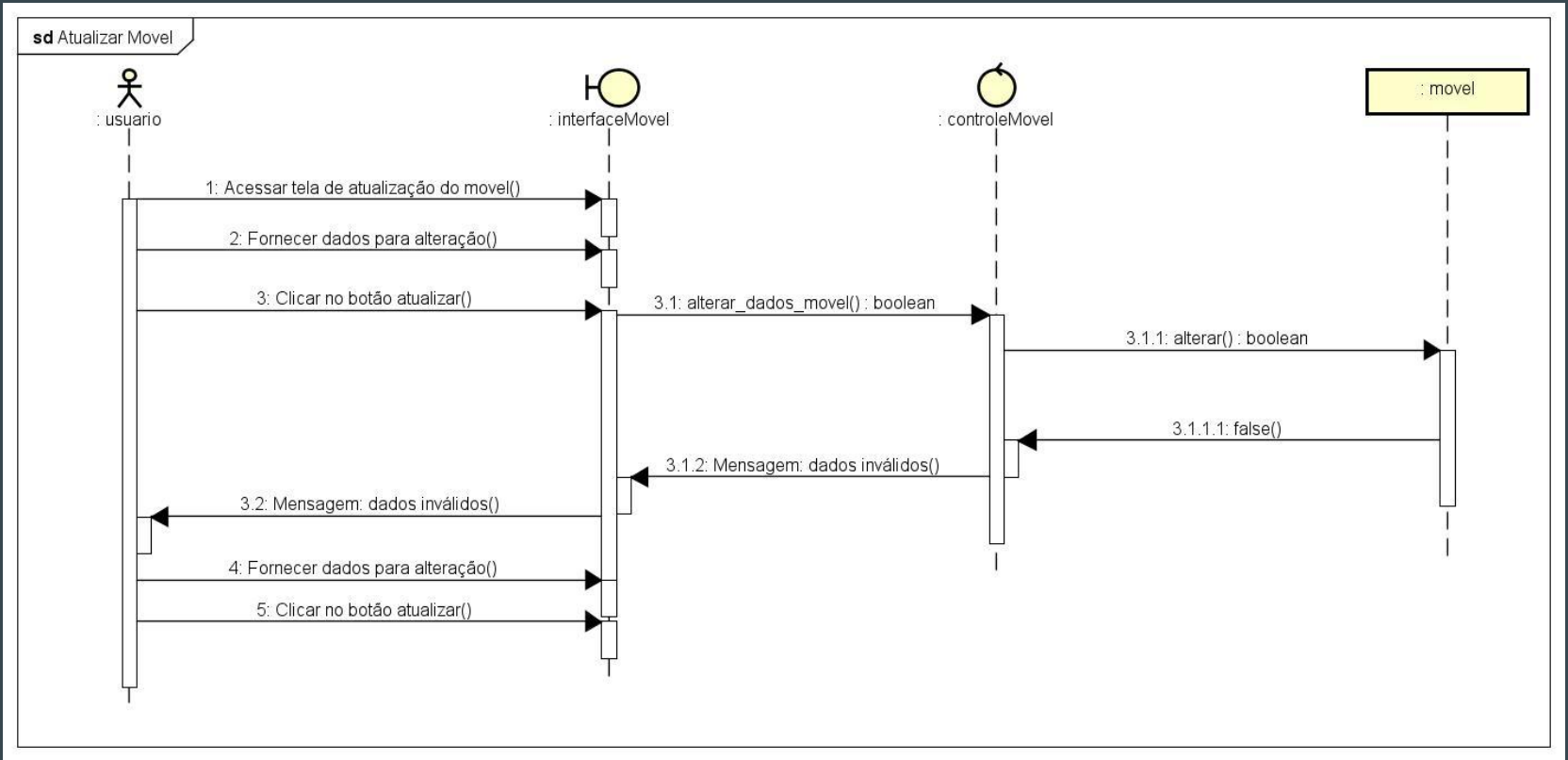
Manter Cômodo - Remover



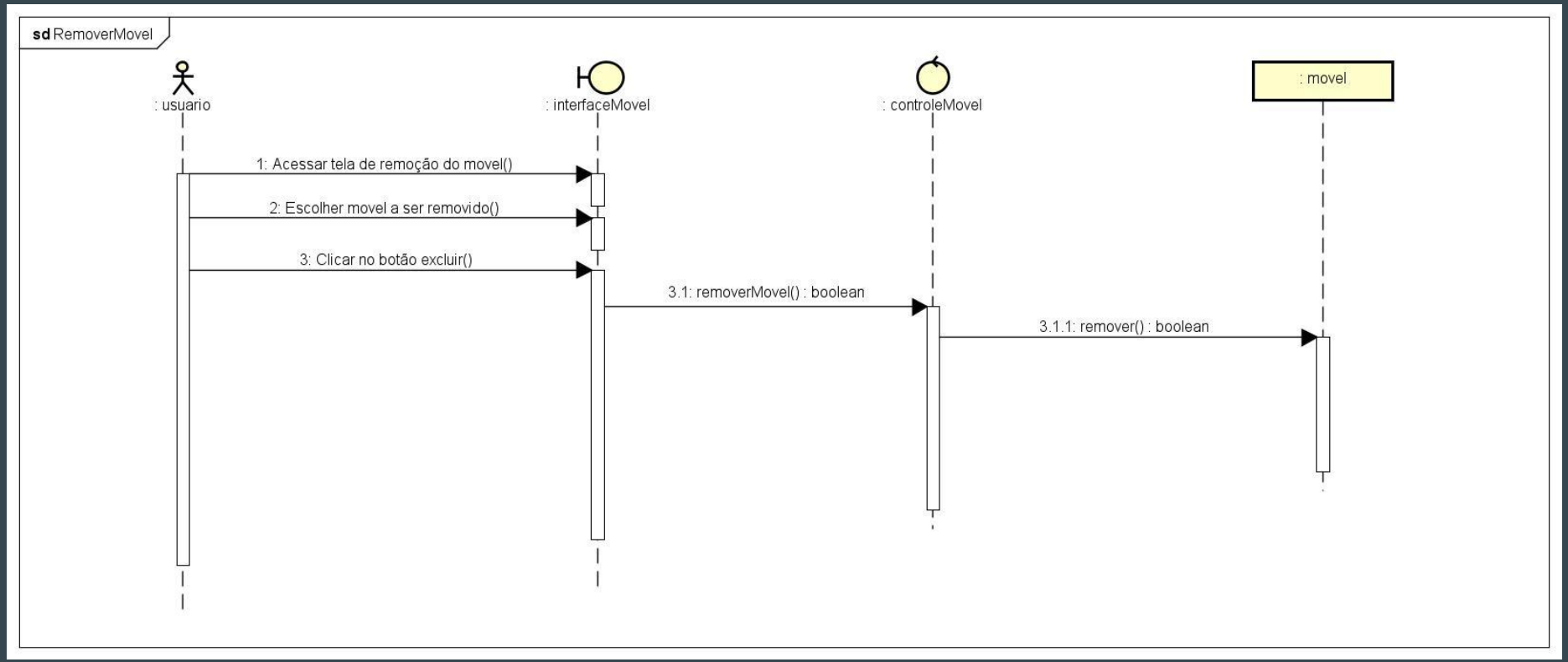
Manter Móvel - Cadastro



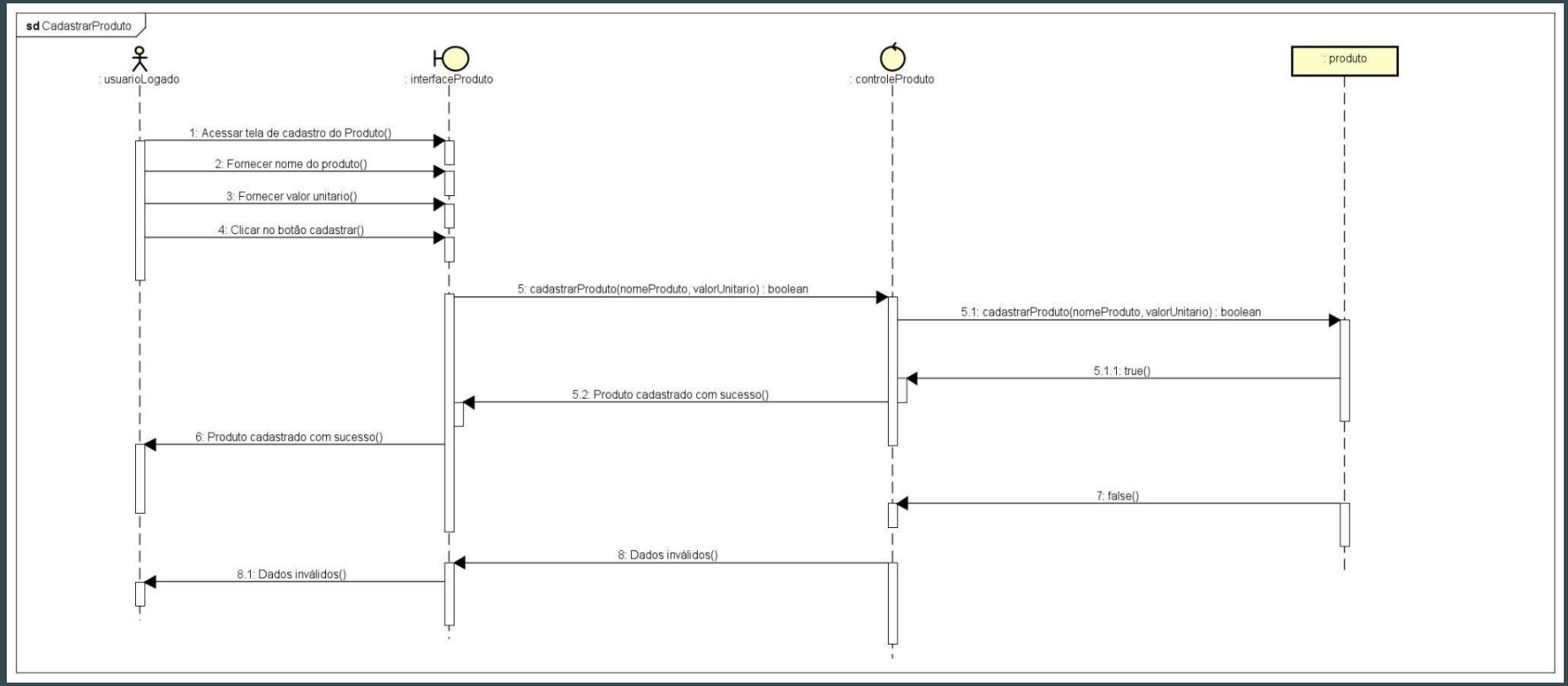
Manter Móvel - Atualizar



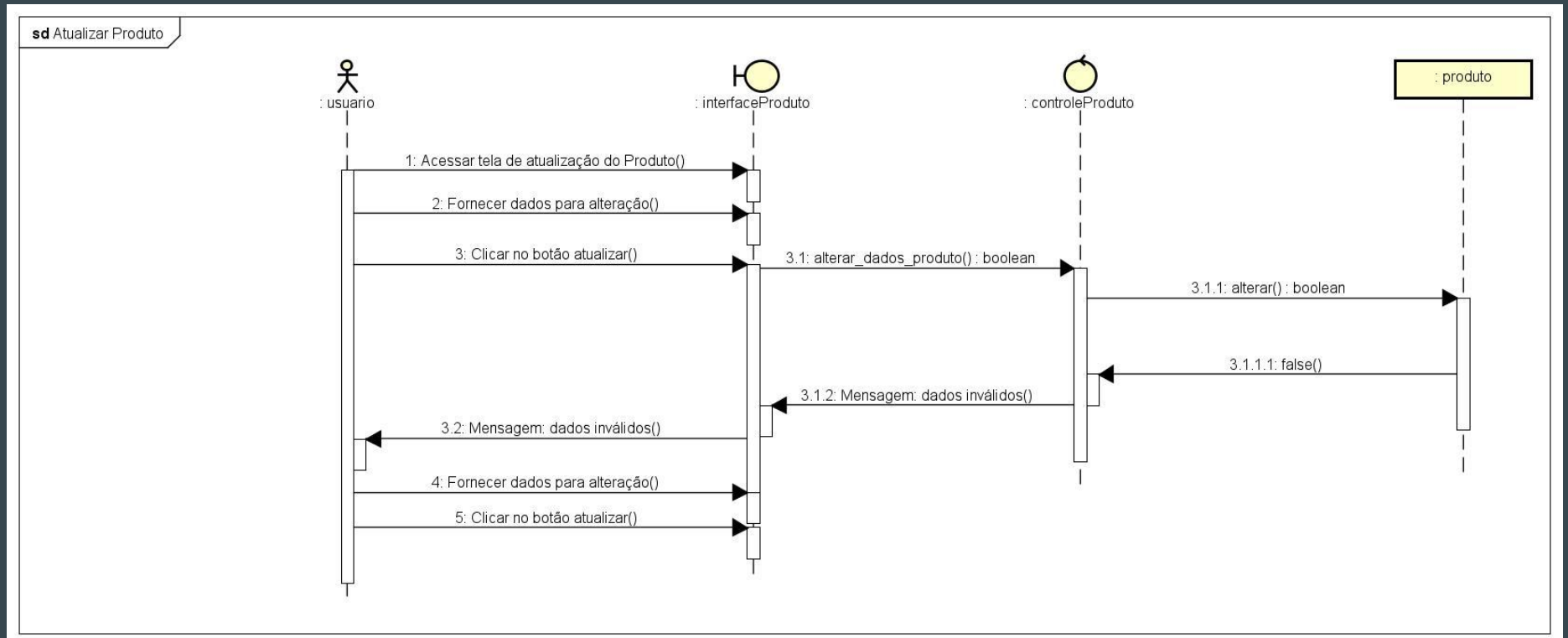
Manter Móvel - Remover



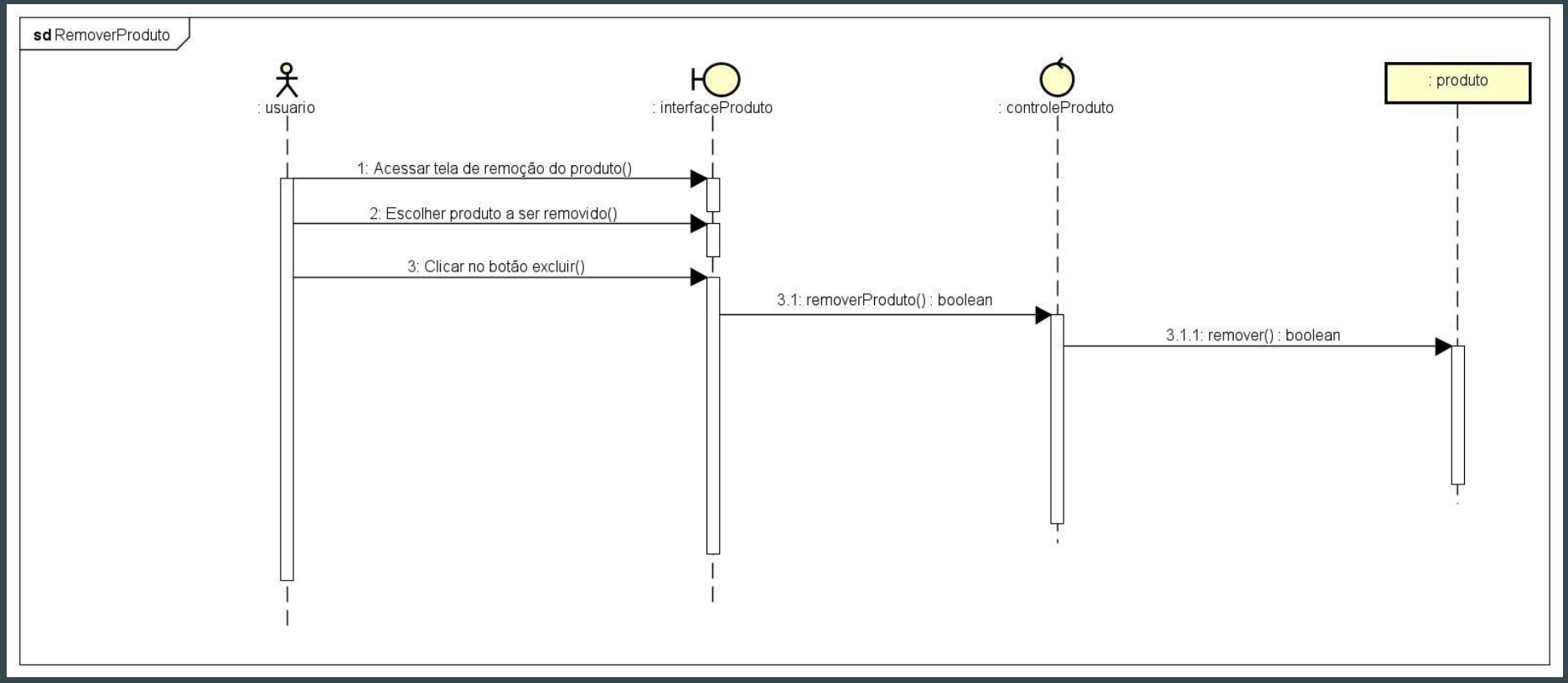
Manter Produto - Cadastro



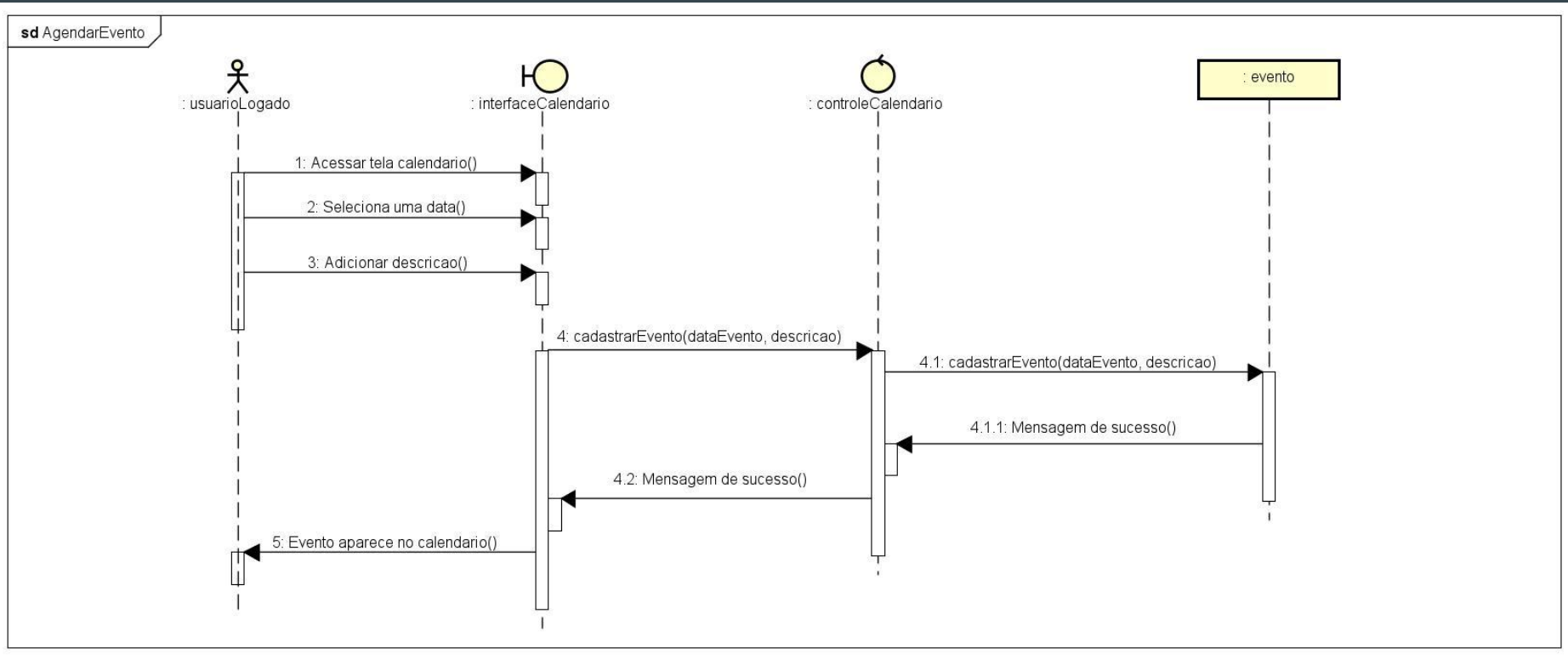
Manter Produto - Atualizar



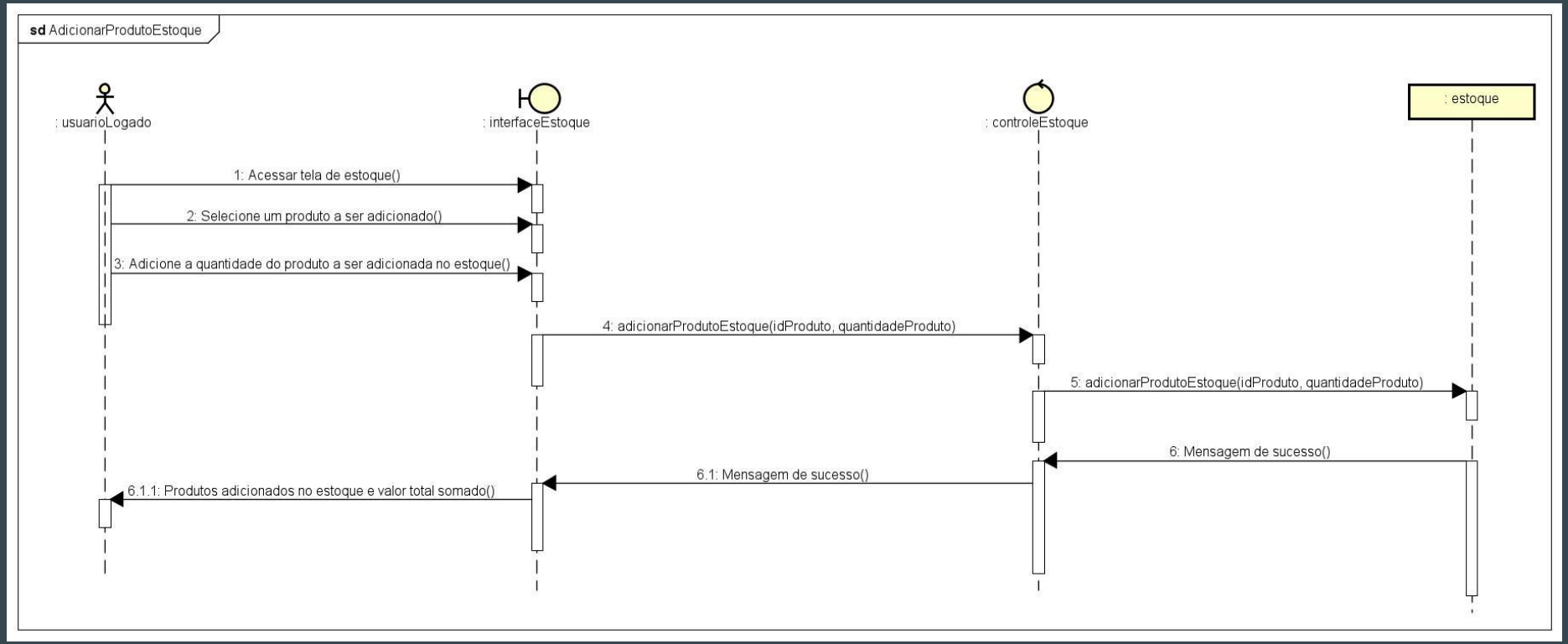
Manter Produto - Remover



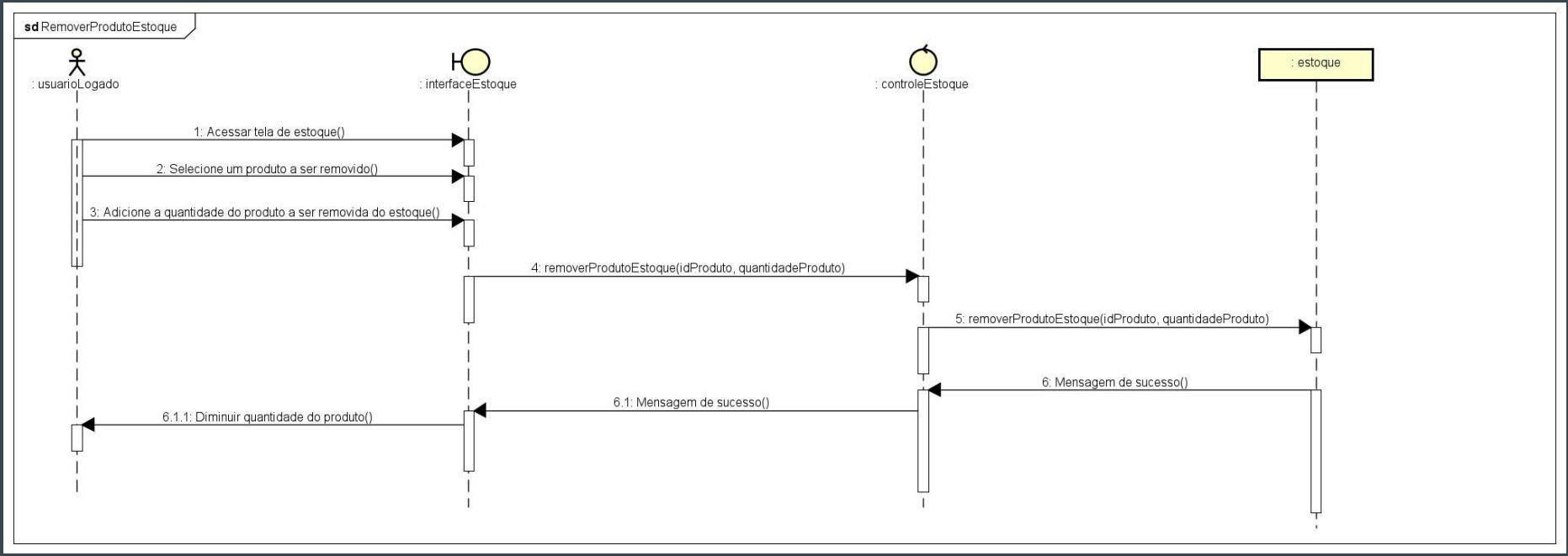
Agendar evento



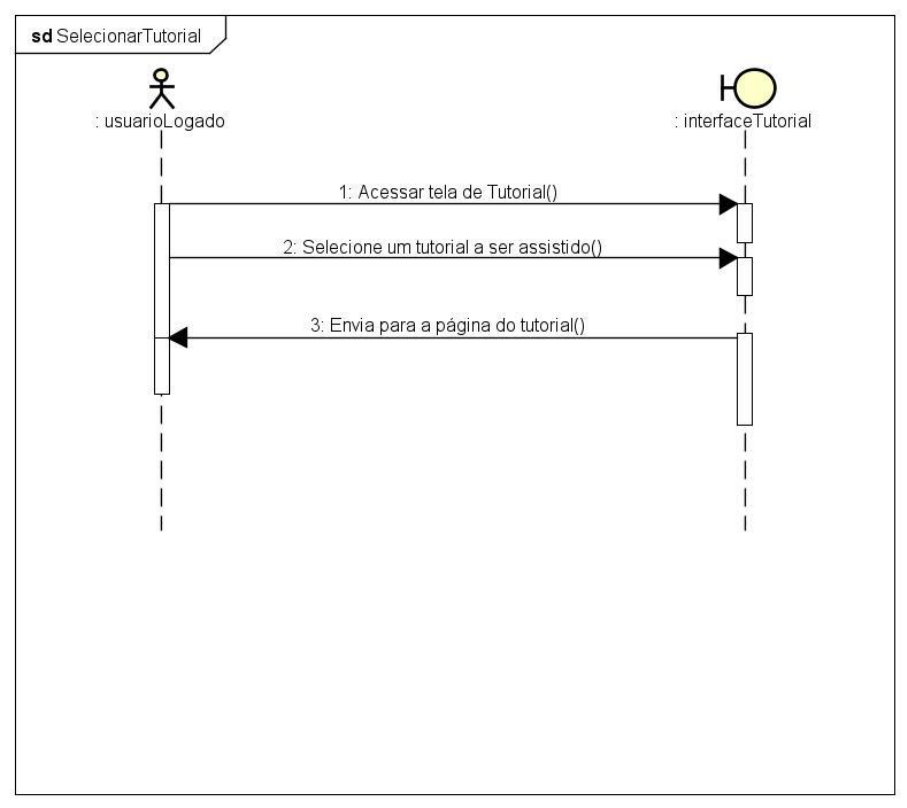
Manter Produto no Estoque - Adicionar Produto



Manter Produto no Estoque - Remover Produto



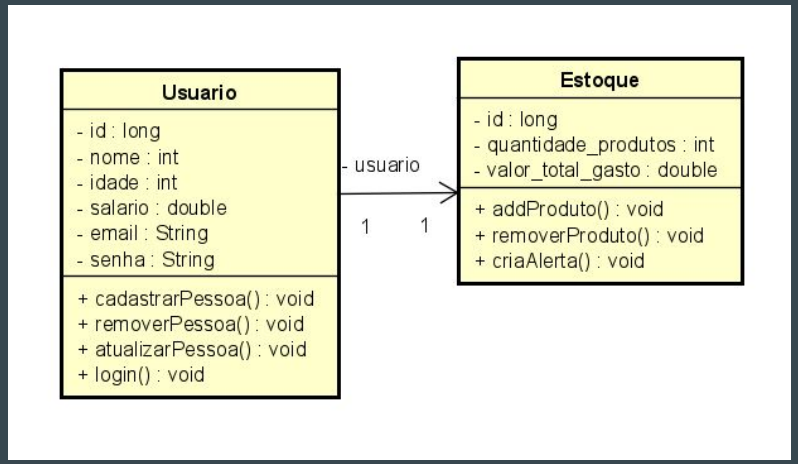
Selecionar Tutorial

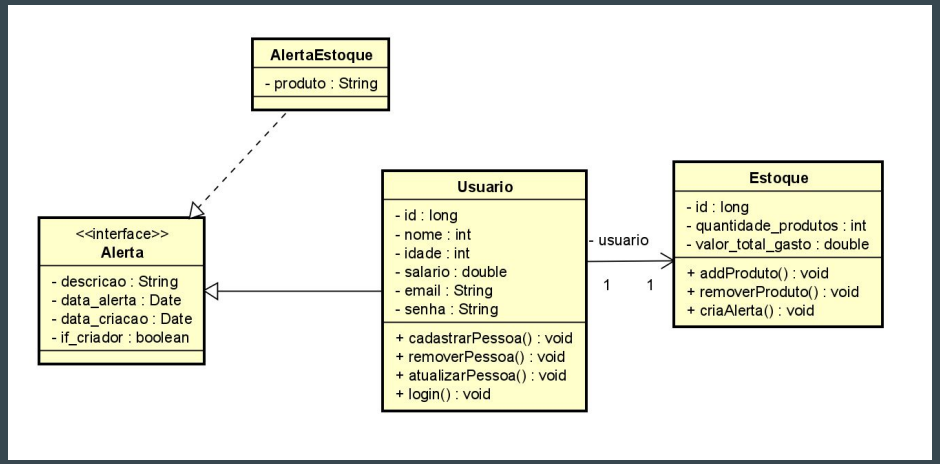


Princípios SOLID: Responsabilidade única

Inicialmente, cada classe do nosso projeto que envolvia o cadastro de alertas possuía um método para tal. Consequentemente, cada uma dessas classes possuía uma responsabilidade a mais.

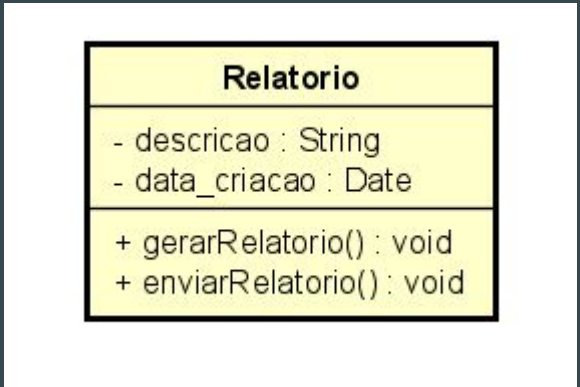
Para evitar esse problema, criamos a classe Alerta, responsável pela criação e gerenciamento de alertas no sistema.

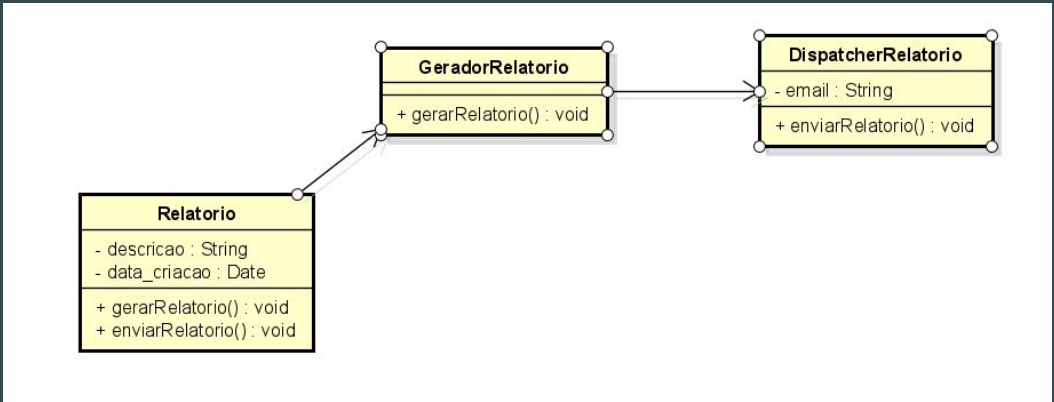




Princípios SOLID: Responsabilidade única

Usamos também esse princípio nas classes de relatório, onde a classe Relatório antes tinha que lidar com gerar o relatório e enviar para o email do usuário no mesmo método. Separamos as responsabilidades de gerar o relatório e enviar o relatório via e-mail pro usuário em dois métodos distintos, em outras classes, mediados pelo método de controle.

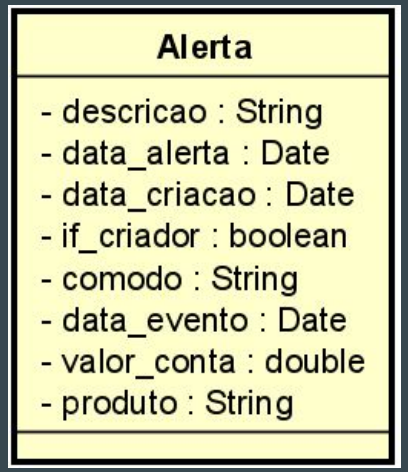


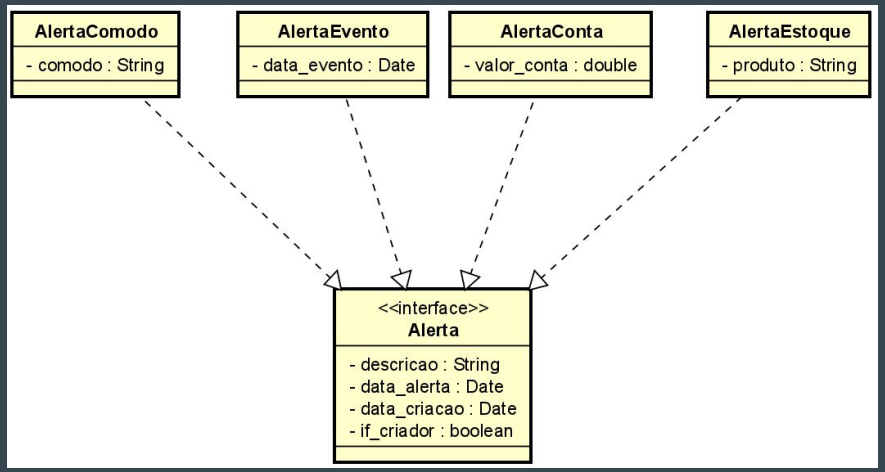


Princípios SOLID: Aberto-fechado

Nosso diagrama de classes previa uma classe alerta, que poderia ser referente ao estado de um cômodo, a uma conta, a um produto no estoque, ou a um evento. A implementação de uma única classe para todos esses casos significaria alterá-la para cada funcionalidade específica de um deles.

Como solução, o Aviso se tornou uma interface, tornando-a aberta para ser implementada, mas fechada para alterações.

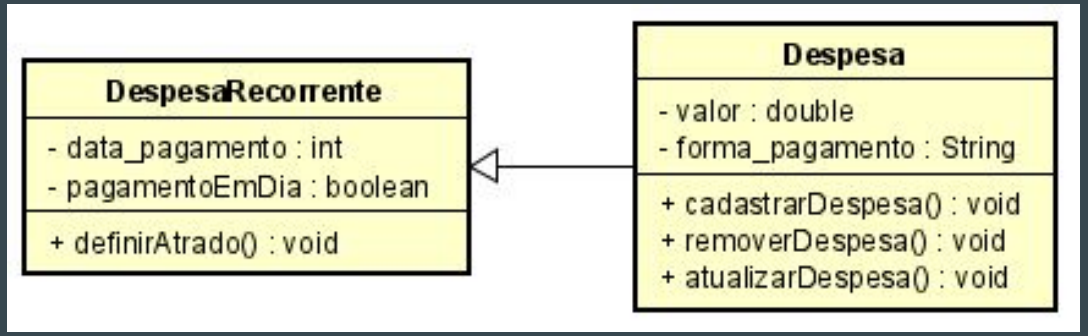




Princípios SOLID: Substituição de Liskov:

A classe “Despesa”, voltada para todos os tipos de despesa do usuário, possuía um atributo referente a contas recorrentes (como conta de luz, água e aluguel), que não faziam sentido para despesas de outras naturezas (como compras de mercado).

Adicionamos, então, uma subclasse voltada para essas despesas recorrentes e colocamos nela o atributo e método pertinente, retirando essa função da superclasse.



Princípios SOLID: Segregação da Interface(ISP)

Foi atendido através da divisão das interfaces conta e cômodo que servem à residência. Ao invés da classe “Residência” atender a uma interface correspondente às duas classes, dividimos em duas classes com suas respectivas interfaces , desta forma cada interface mantém apenas o escopo de interesse da classe correspondente

Princípios SOLID: Inversão de dependências

Para aplicar o princípio de inversão de dependências, foi criada a interface Alerta, que vai ser passada para o método do usuário como tipo de parâmetro. Deste modo, a classe só deve se preocupar com o contrato firmado, e não com a implementação de cada tipo de classe, por consequência, não precisaria criar vários métodos "redundantes", um para cada classe, criando um código mais flexível e duradouro.