# Sistema de caixa de loja

Integrantes:
Henrique Neves Braga
Fellipe Carvalho Pombo Lima

## Escopo do sistema:

O sistema corresponde a aplicação de um caixa de loja permitindo o gerenciamento, que se entende por, cadastro, alteração, remoção e consulta de entidades relevantes a logística do caixa, sendo essas: funcionários, clientes, vendas, descontos e produtos

O sistema deve também restringir o gerenciamento de certas entidades para os gerentes exclusivamente, fazendo a autenticação do login e a distinção de usuários. Além disso, deve permitir que seja feito o relatório de vendas de um mês.

## Restrições:

- O sistema só pode ser acessado pela rede local
- O sistema deve estar disponível para uso durante todo o horário de funcionamento da loja
- O sistema deve ser fácil, simples e intuitivo
- O sistema deve ser capaz de atender lojas de diferentes segmentos

## Requisitos Arquiteturais:

- O sistema deve exigir autenticação para ser usado
- O sistema deve ser baseado em web
- O sistema deve ser portátil
- O sistema deve suportar múltiplos funcionários acessando-o ao mesmo tempo
- O sistema deve permitir que funcionários criem, removam, alterem e consultem a tabela de vendas do banco de dados
- O sistema deve garantir que apenas funcionários do tipo gerente possam cadastrar. remover funcionários e alterar funcionários
- O sistema deve garantir que apenas funcionários do tipo gerente cadastrem, alterem e removam descontos

## Definição dos padrões de arquitetura:

Considerando a necessidade de um sistema distribuído, onde vários terminais de caixa conectam-se a um servidor central para acessar os dados e funcionalidades, um padrão arquitetural adequado seria o Modelo Cliente-Servidor.

Este modelo permite uma divisão clara entre o cliente (o terminal do caixa) e o servidor (que gerencia o banco de dados e processa as operações).

# Justificativa das decisões:

obs.: seguimos o modelo de qualidade ISO 25010

Portabilidade:

Para garantir a portabilidade do sistema em máquinas diversas de caixa, o sistema front-end é baseado em web usando apenas javascript, css e html, onde o browser garante a portabilidade.

Confiabilidade:

-Disponibilidade - O modelo Cliente/Servidor mantém o sistema operacional enquanto o Servidor estiver operando, dessa forma mesmo se apenas uma pessoa quiser acessar, o sistema estará ativo, pois o servidor será mantido ativo.

Segurança:

-Autenticidade: Garantido pelo sistema backend e o banco de dados, o login e a senha do usuário são verificados no sistema servidor para o acesso ser permitido.

-Responsabilidade: A responsabilidade é garantida pela associação de um funcionário a toda venda registrada no banco de dados, possibilitando rastreá-lo a venda.

-Integridade: A diferenciação entre gerente e funcionário será feita pelo backend e pelo banco de dados, garantindo que somente o gerente tenha acesso a algumas funcionalidades

# Princípios SOLID:

Singles Responsibility: Cada classe cuida de uma área específica.

- Caixa, cuida do cadastro de vendas e do relatório de vendas.
- Funcionário pode consultar produtos e cadastrar e remover clientes.
- Gerente, fora o papel de funcionário, pode também realizar os CRUDs de funcionário e produto.
- GerenciadorLogin é responsável pela lógica de login
- ConstrutorVenda pela lógica da construção de um objeto “venda”.
- Produto e Venda guardas informações sobre produto e venda respectivamente

Open-Closed: Novas funcionalidades não precisam modificar métodos existentes, os métodos atuais de cada classe tem métodos claros que fazem coisas específicas. Uma nova funcionalidade poderia usar esses métodos, mas não precisaria modificá-los.

Liskov-Substitution: O único caso de herança é quando Gerente herda funcionário e ele pode fazer tudo o que Funcionário pode, respeitando a substituição de Liskov.

Interface Segregation: Não há classes que implementam interfaces com métodos que elas não usam.

Inversão de dependência: As classes não dependem de outras classes mas sim de interfaces.

## Padrões GRASP:

Information expert: Classes como Venda e Produto fornecem informações sobre si mesmos

Creator: A classe Facade que é a única que utiliza Funcionario/Gerente, Caixa, GerenciadorLogin e ConstrutorVenda é responsável por criá-las.

Baixo acoplamento: Com o uso de interfaces as classes não dependem da implementação uma das outras.

Alta coesão: As classes possuem responsabilidades relacionadas entre si.

## Padrões GoF:

Facade: Para fornecer uma interface única e simplificada para o acesso às funcionalidades das classes Caixa, Funcionário e Gerente, ConstrutorVenda e GerenciadorLogin. Quando um botão é apertado a classe de interface gráfica chama o respectivo método do facade que por sua vez chama as classes adequadas. Dessa maneira não é preciso conhecer os detalhes da implementação interna para criar a interface.

Observer: Um objeto da classe ConstrutorVenda é responsável por construir o objeto venda ao mesmo tempo que o usuário adiciona e remove os produtos na interface. Quando o usuário termina de ajeitar os produtos e clica em cadastrar, usando o padrão observer, o objeto construtor de venda avisa ao caixa para que ele possa atualizar o bancos de dados. Nesse caso o caixa observa o construtor de vendas.

Singleton: Para garantir que exista apenas uma instância da classe Caixa em todo o sistema. Isso centraliza a lógica de registro de vendas e geração de relatórios, evitando redundâncias e inconsistências.

## Modificações para atender aos princípios:

Primeiramente agora toda classe implementa uma interface, diferente da primeira ideia do projeto. Isso foi para reduzir o acoplamento.

Mais uma medida importante foi o uso do padrão GoF facade que não era originalmente planejado. Isso permitiu não haver acoplamento entre a interface gráfica e a regra de negócio. A interface gráfica usa métodos do Facade quando o usuário clica em algum botão independentemente da classe que irá tratar as informações passadas pelo usuário.

Foi criada uma classe para gerenciar o login. Isso é importante para permitir que diferentes empresas usem sua própria lógica de login.

A criação de uma classe responsável por construir venda conforme o usuário adiciona e

remove itens na interface é importante para o caso de futuramente se desejar alguma funcionalidade que exija manipular os dados de uma venda antes dela ser concluída.

A criação de uma classe para gerenciar a interface gráfica foi uma das decisões mais importantes para que a parte visual seja facilmente modificável sem precisar interagir com a lógica mais profunda do programa.

## A evolução do projeto para uma LPS:

Mandatórias:

- Cadastro e login de funcionários
- Registro de vendas
- Controle de estoque
- Sistema de pagamento
- Sistema Reembolso

Opcionais:

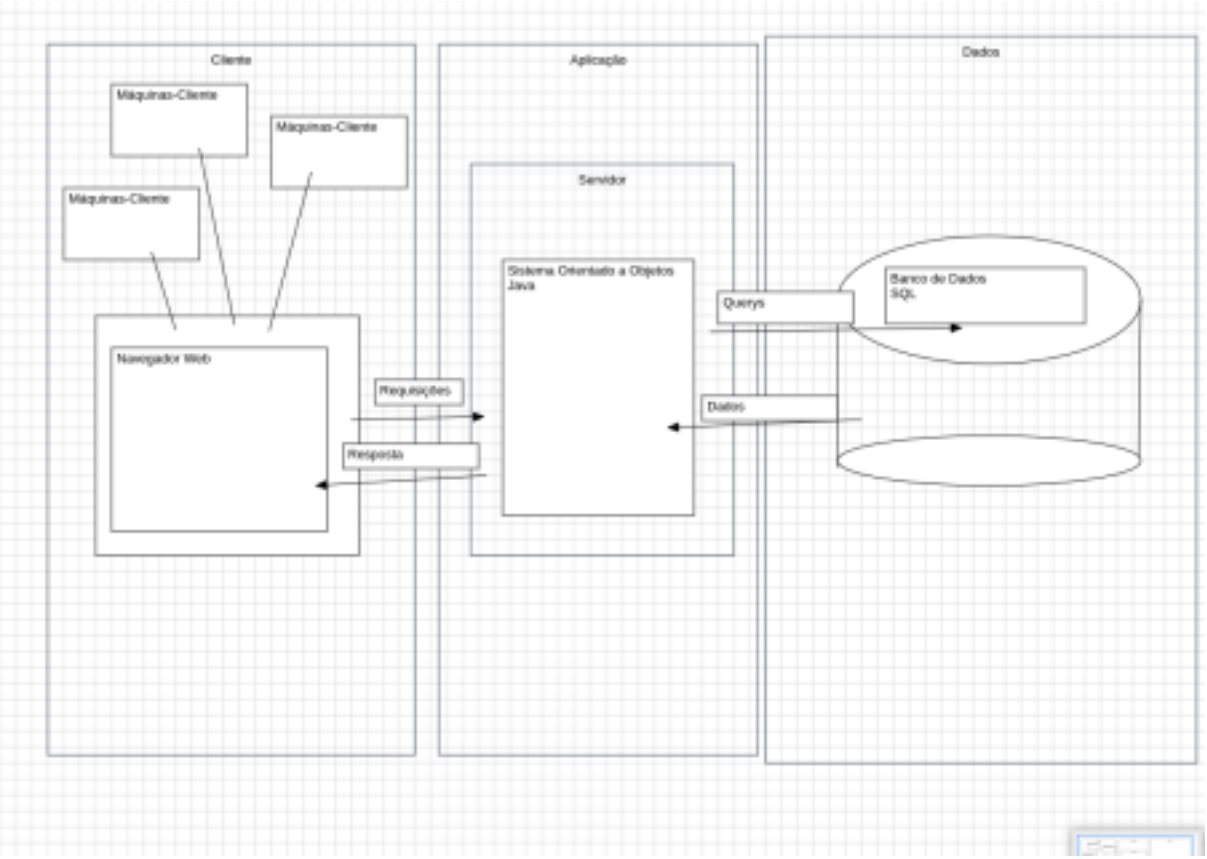
- Sistema de fidelidade de clientes
- Sistema de desconto para produtos específicos
- Gerenciar funcionários, produtos e descontos
- Relatório de vendas

Alternativas:

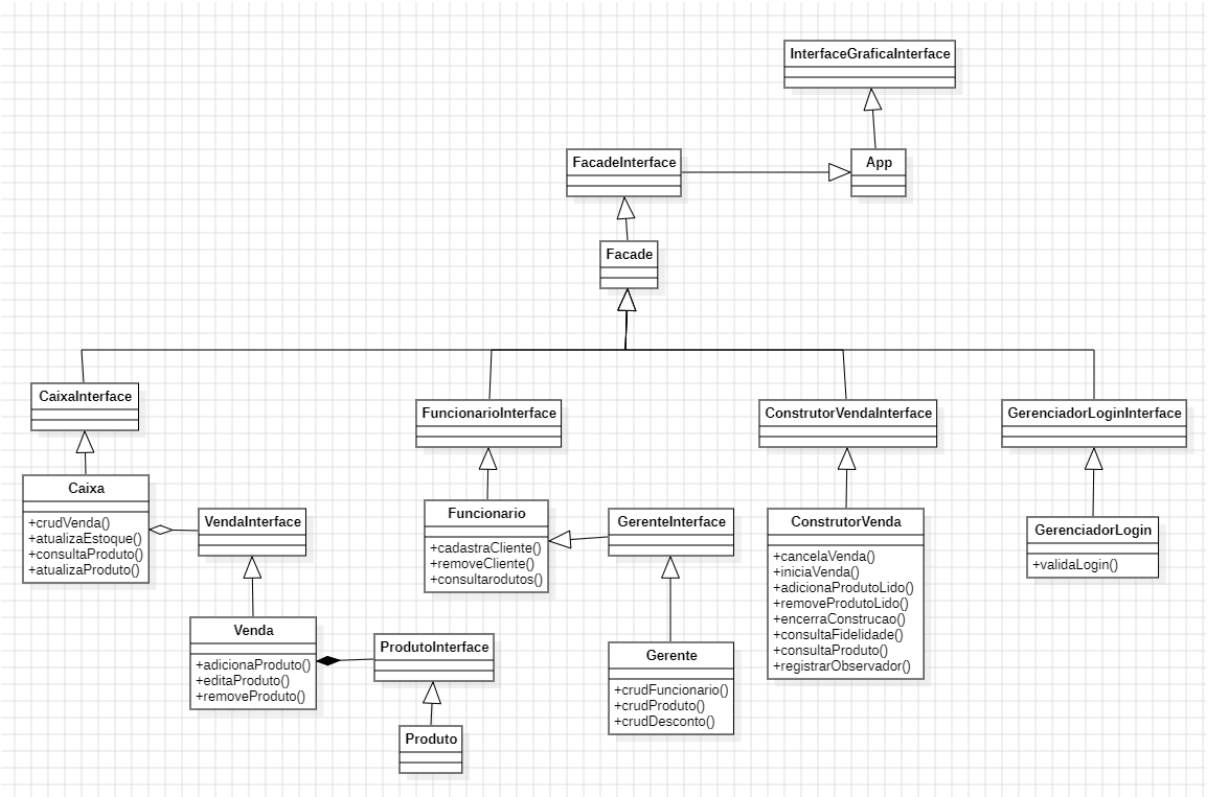
- Controle de estoque com validade para produtos perecíveis (supermercados, farmácias).
- Gerenciamento de mesas e pedidos com divisão de conta para restaurantes e bares.

# DIAGRAMAS

## Diagrama de arquitetura geral:



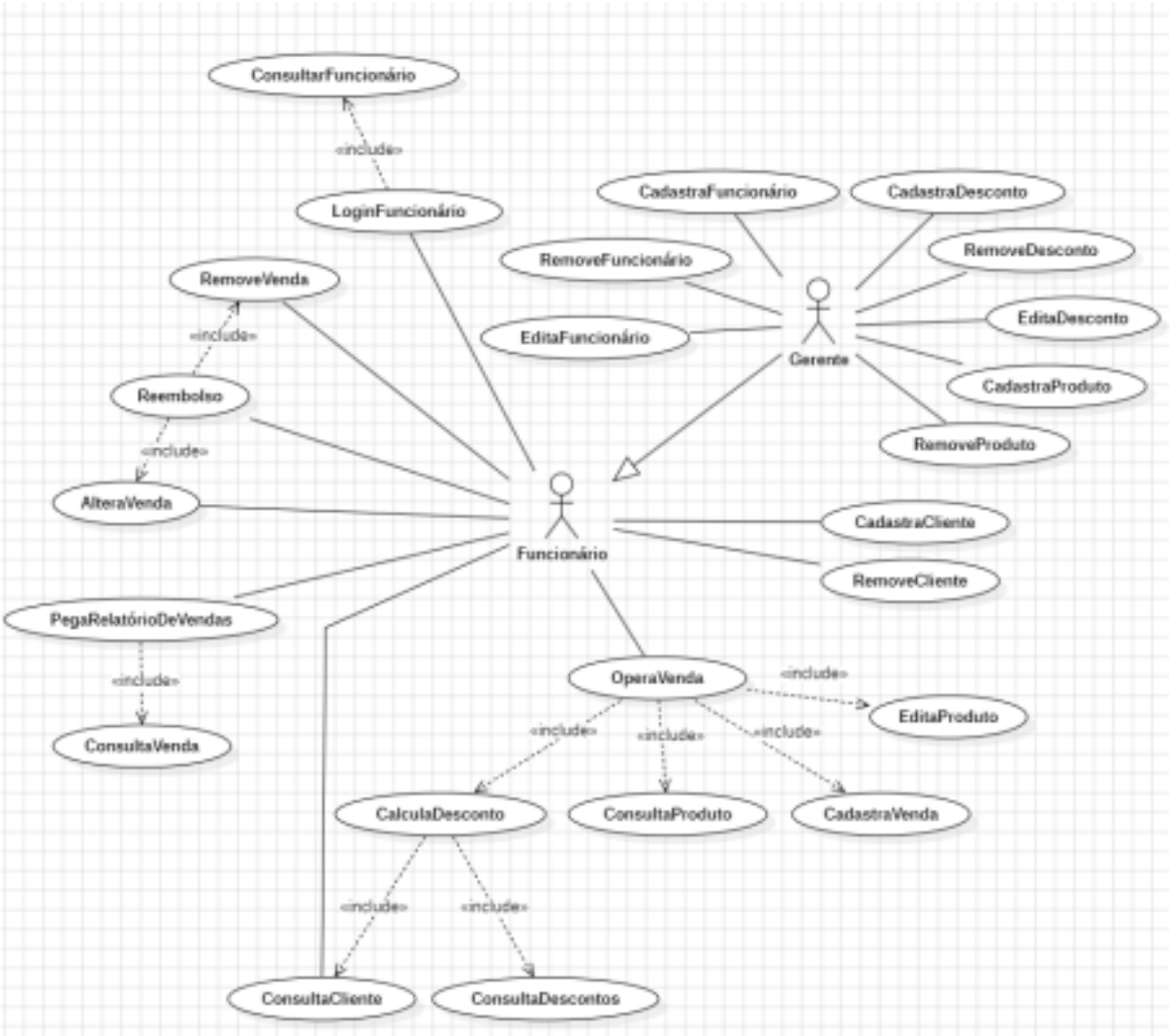
# Diagrama de classe:



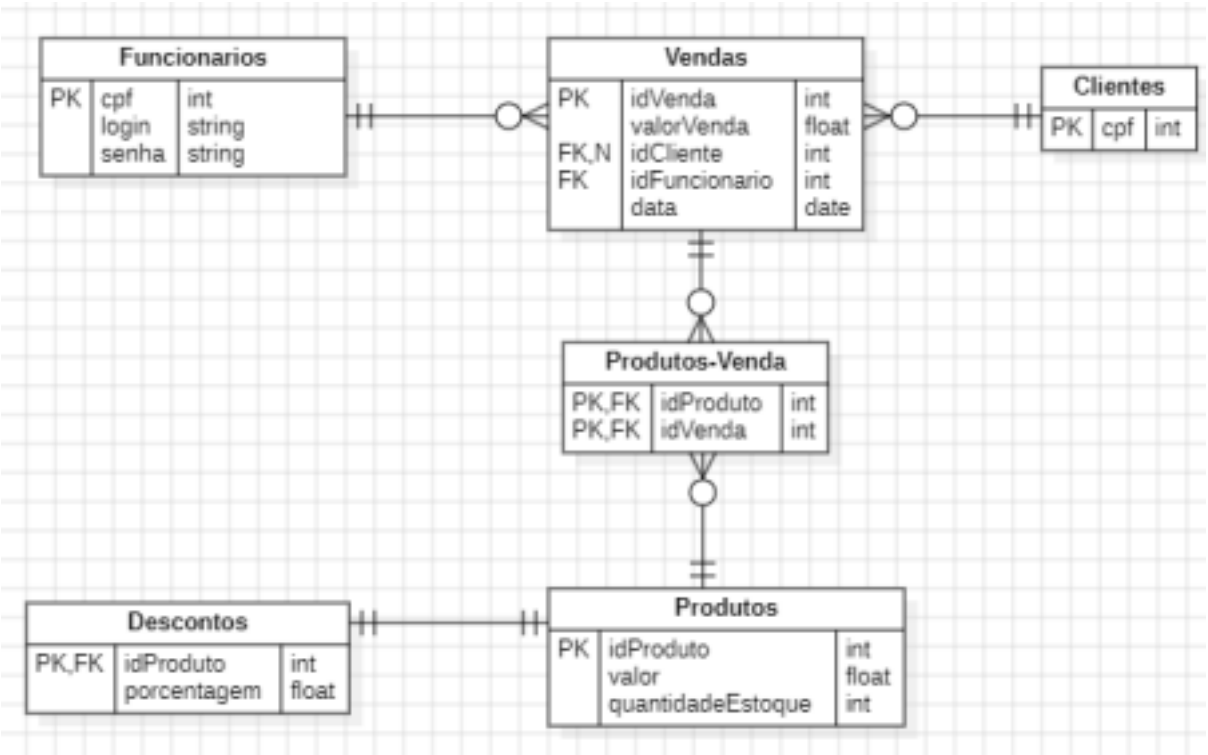

Informações foram omitidas para que o diagrama coubesse na página de forma legível. A classe App possui os métodos para criar a interface. A classe facade possui métodos para acessar os métodos das classes que implementam as interfaces CaixaInterface, FuncionarioInterface (ou GerenteInterface), ConstrutorVendaInterface e GerenteLoginInterface.

As classes que terminam com interface representam interfaces que são implementadas pelas classes abaixo delas.

## Diagrama de casos de uso:

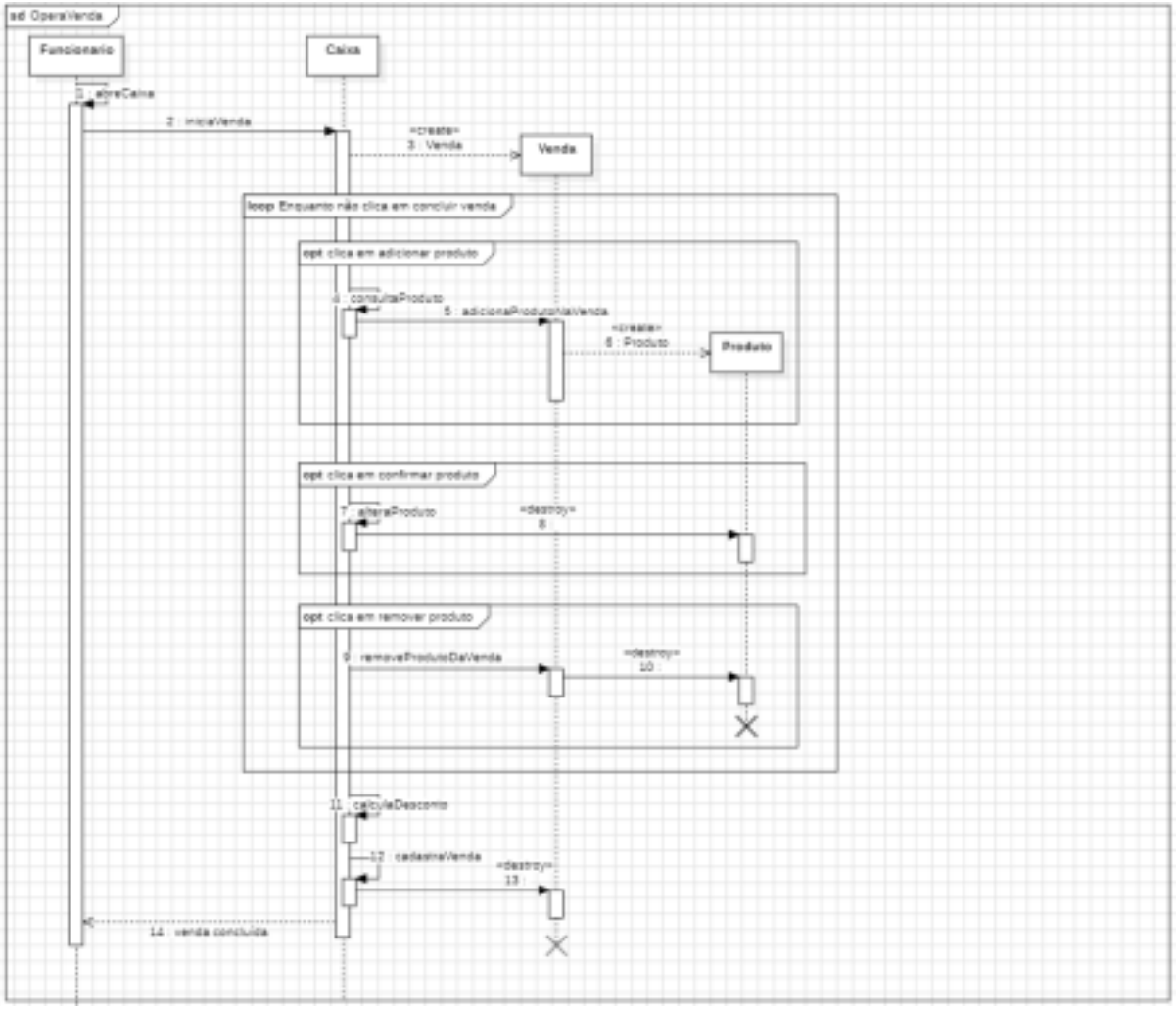


## Modelo conceitual:

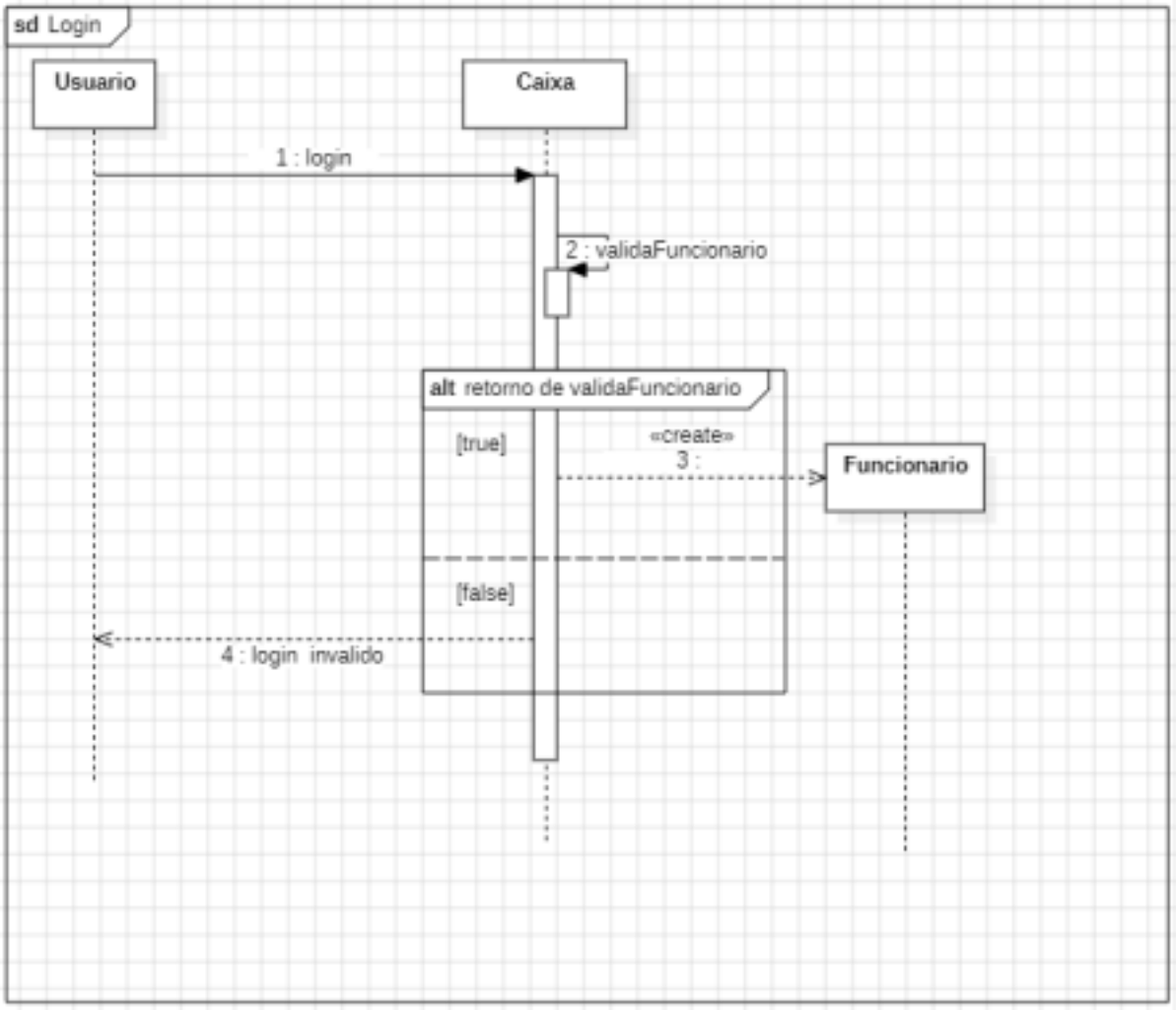


# Diagramas de sequência:

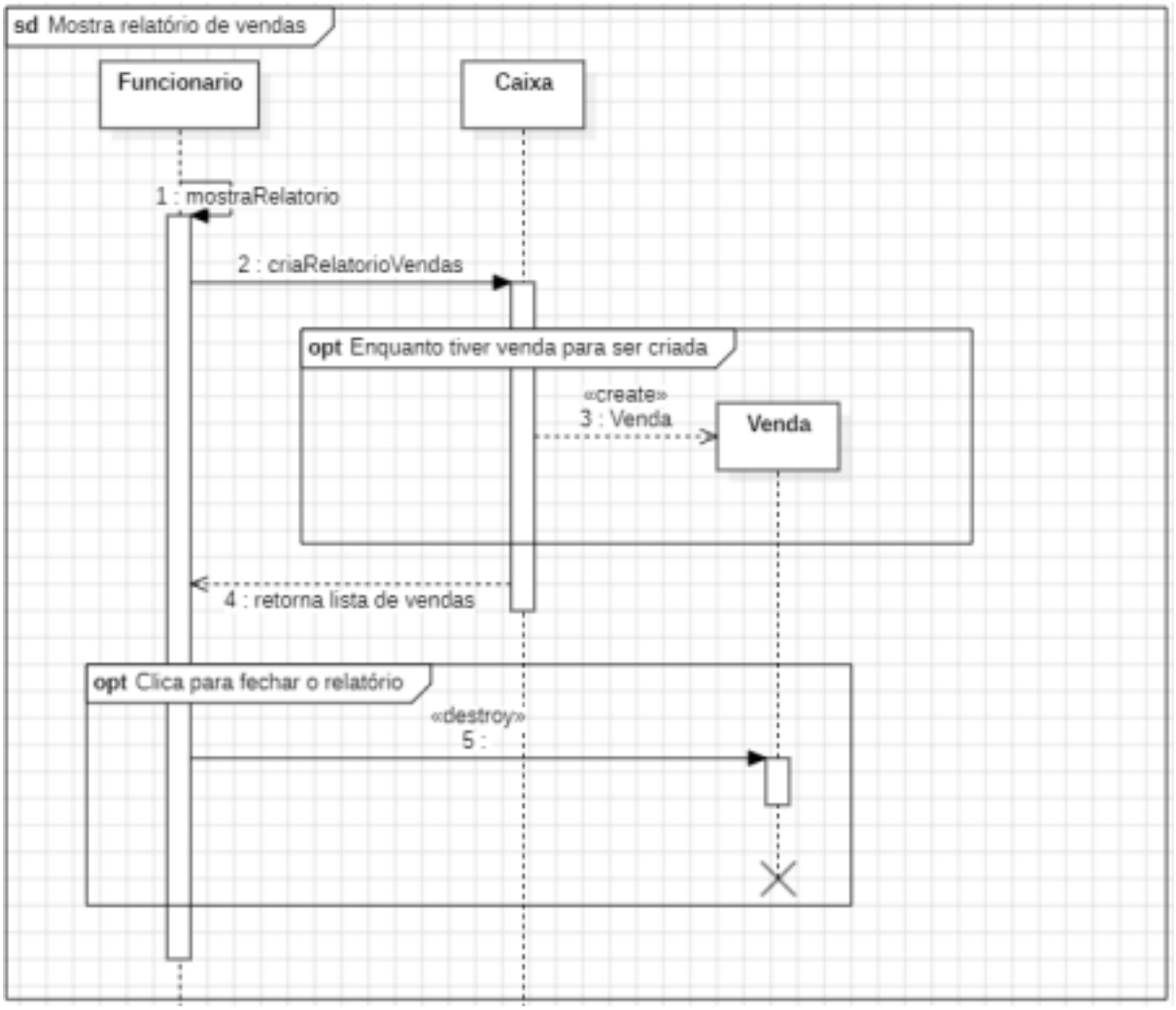
OperaVenda:



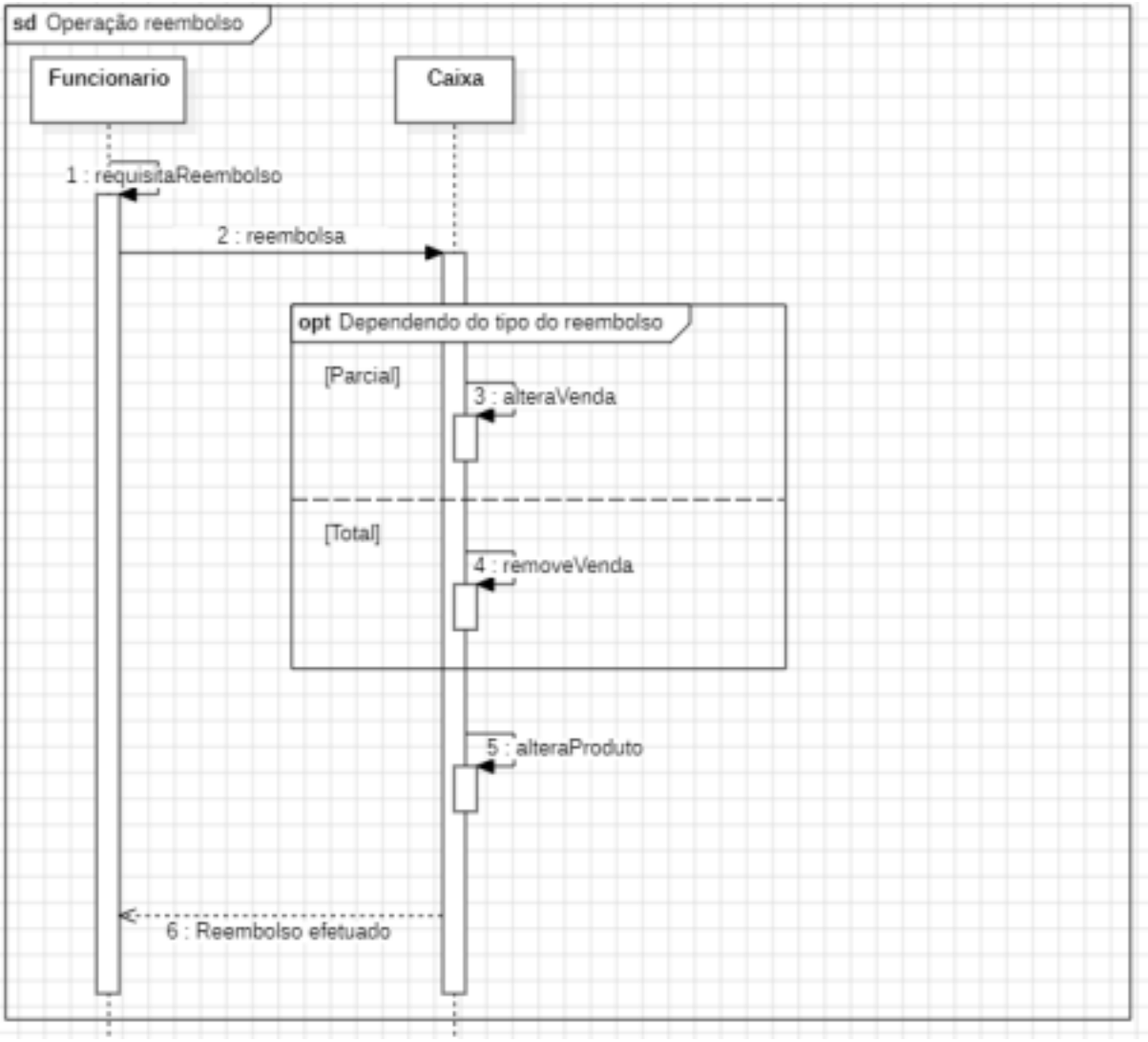
Login:


sd Login
Usuario
Caixa
1 : login
2 : validaFuncionario
alt retorno de validaFuncionario
[true]
«create»
3 :
Funcionario
[false]
4 : login invalido

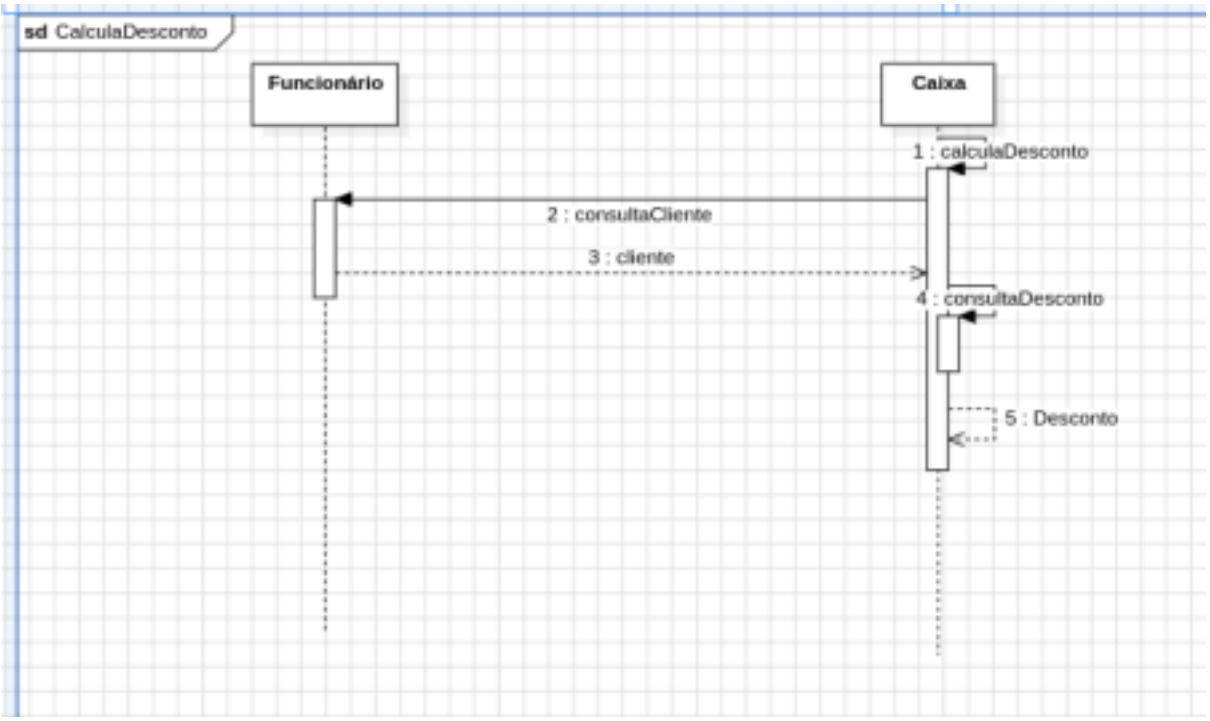
Relatório:


sd Mostra relatório de vendas
Funcionario
Caixa
1 : mostraRelatorio
2 : criaRelatorioVendas
opt Enquanto tiver venda para ser criada
«create»
3 : Venda
Venda
4 : retorna lista de vendas
opt Clica para fechar o relatório
«destroy»
5 :

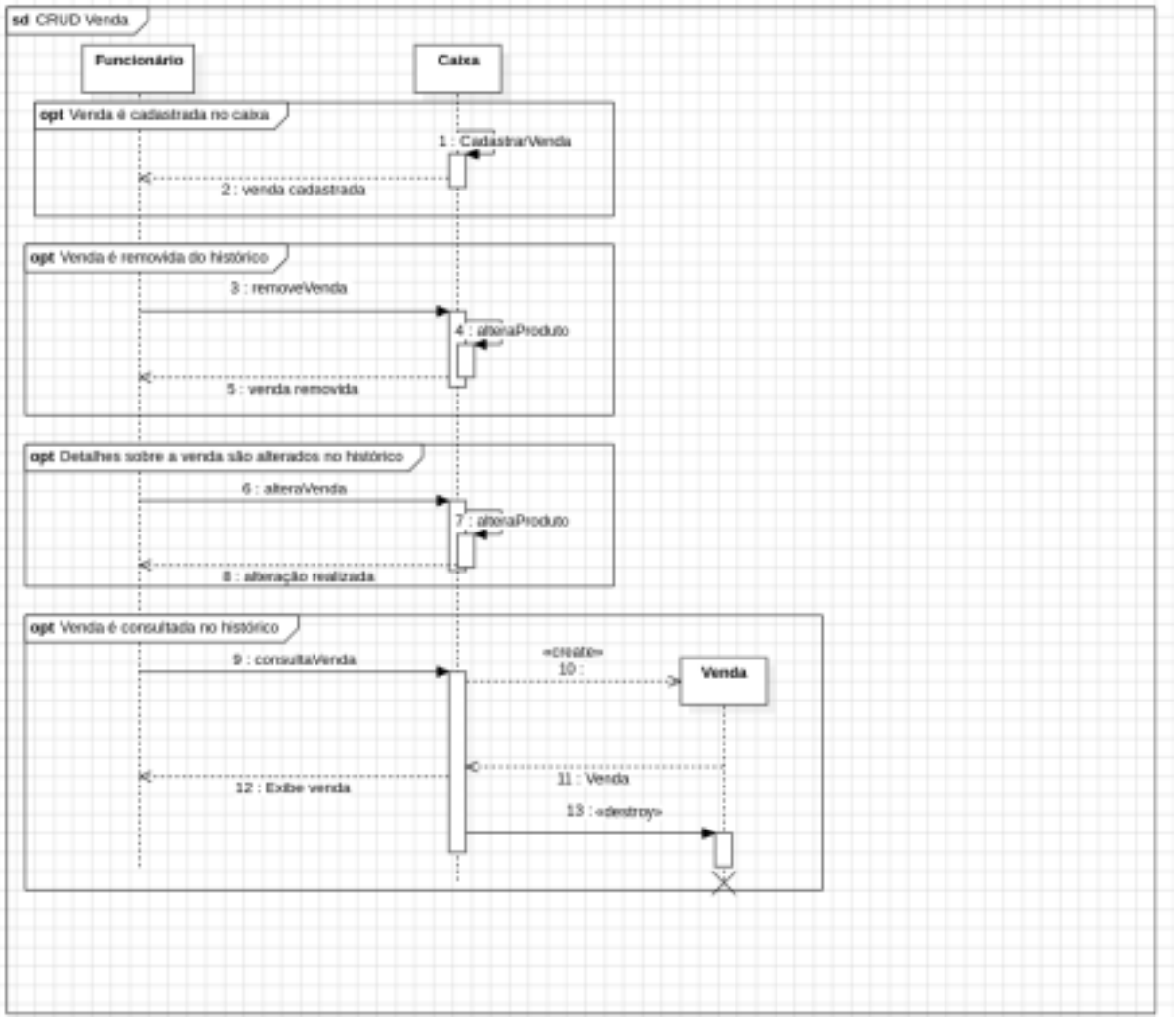
Reembolso:


sd Operação reembolso
Funcionario
Caixa
1 : requisitaReembolso
2 : reembolsa
opt Dependendo do tipo do reembolso
[Parcial]
3 : alteraVenda
[Total]
4 : removeVenda
5 : alteraProduto
6 : Reembolso efetuado

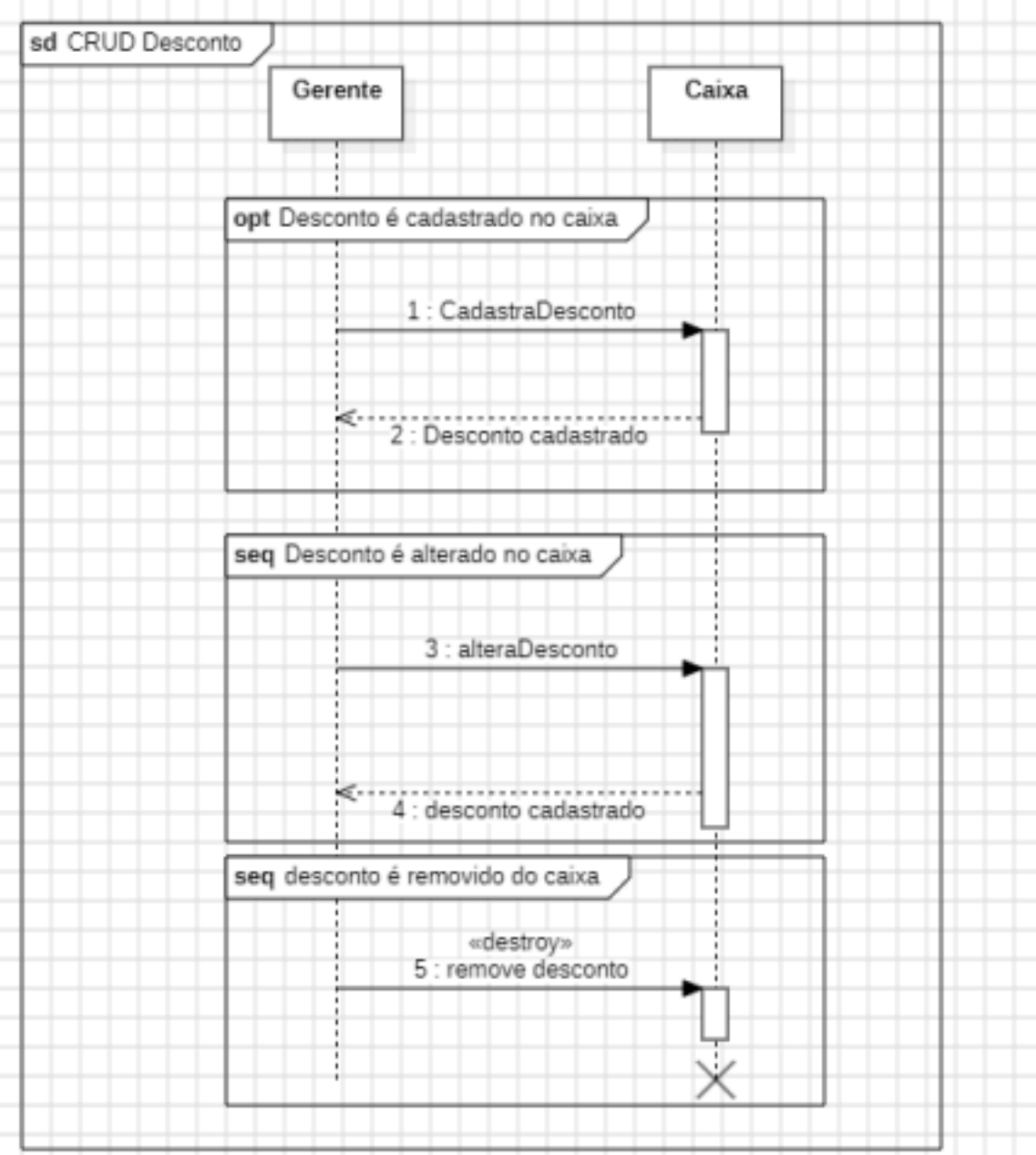
CalculaDesconto:



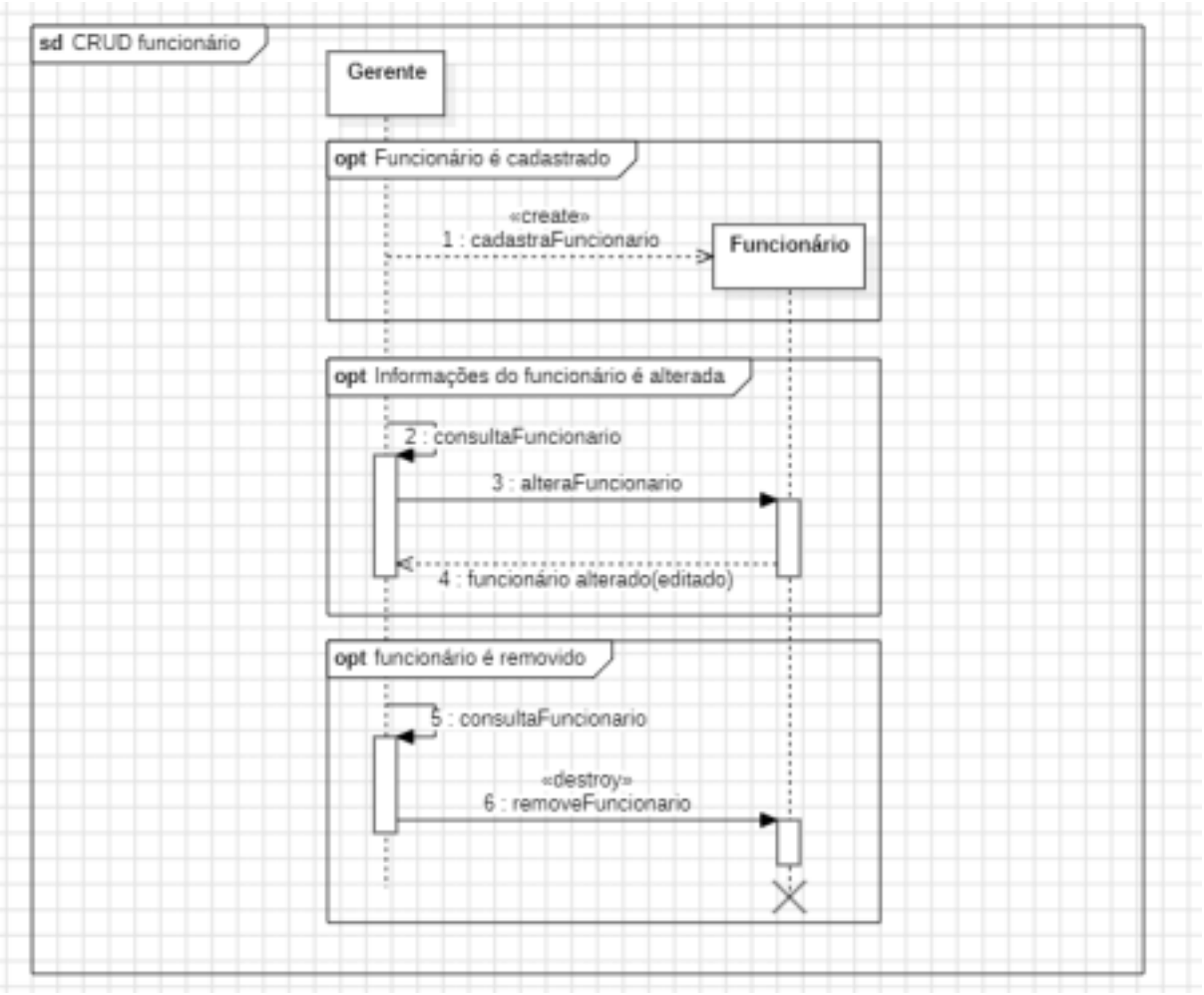
CRUD Venda:


sd CRUD Venda
Funcionário
Caixa
opt Venda é cadastrada no caixa
1 : CadastrarVenda
2 : venda cadastrada
opt Venda é removida do histórico
3 : removeVenda
4 : alteraProduto
5 : venda removida
opt Detalhes sobre a venda são alterados no histórico
6 : alteraVenda
7 : alteraProduto
8 : alteração realizada
opt Venda é consultada no histórico
9 : consultaVenda
«create»
10 :
Venda
11 : Venda
12 : Exibe venda
13 : «destroy»

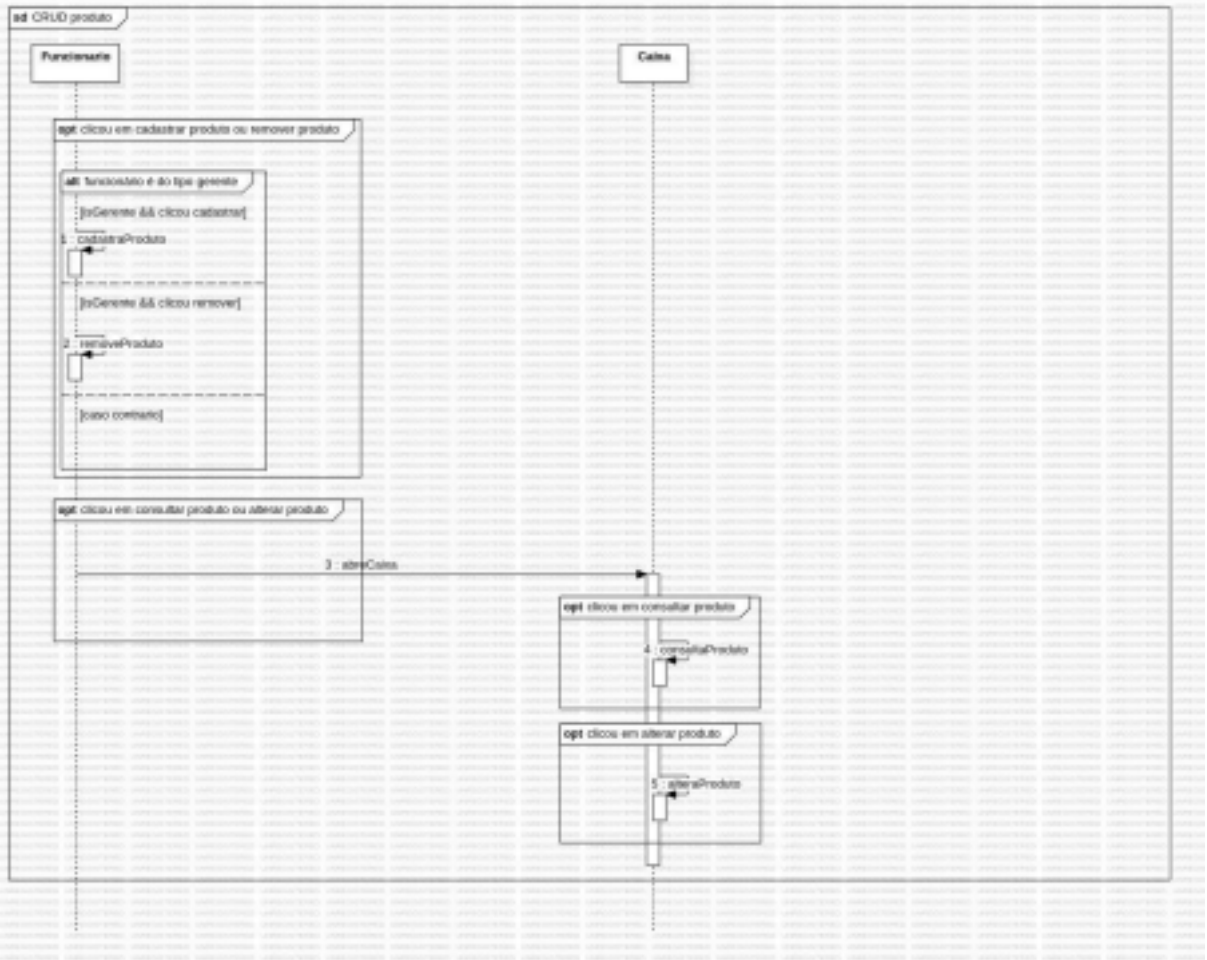
CRUD Desconto:



CRUD Funcionário:



CRUD produto



CRUD clientes