UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Sistema de Vendas

Projeto de Software

Componentes do Grupo e Matrícula

- **Ana Luisa Esposito - 217083089**
- **Gabriel Henrique Brandão Corrêa Silva - 119083069**
- **José Emiliano Bertholdo Júnior - 217083125**
- **Lucas Rogério Silva Ferreira - 217083103**
- **Rodrigo Werneck Nogueira de Medeiros - 216083099**

Links Github

- **Frontend - https://github.com/vvrnck/uff-pdv-frontend**
- **Backend - https://github.com/lucasrrogerio/vendas-sys**

2020

Sumário

1. Introdução

Este documento apresenta a especificação dos requisitos para o desenvolvimento do software de gerenciamento de vendas do mercado Comércio Sinistro. Ele fornece aos desenvolvedores as informações necessárias ao projeto e sua implementação.

O código da aplicação está disponível no gitHub, o backend aqui, com instruções de como rodar localmente a aplicação, e frontend aqui.

1.1 Objetivos

O documento tem como objetivo fornecer uma visão geral de arquitetura do sistema. Foram utilizadas várias visões arquiteturais para representar os diversos aspectos da plataforma oferecida. Pretendemos transmitir nossas decisões arquiteturais que foram chaves para o desenvolvimento do sistema.

1.2 Escopo

Este documento fornece uma visão geral da arquitetura do sistema. A plataforma Venda Sys está sendo produzida pela equipe de desenvolvedores para dar suporte nas operações do cliente, auxiliando na execução das atividades de vendas e registros de compras.

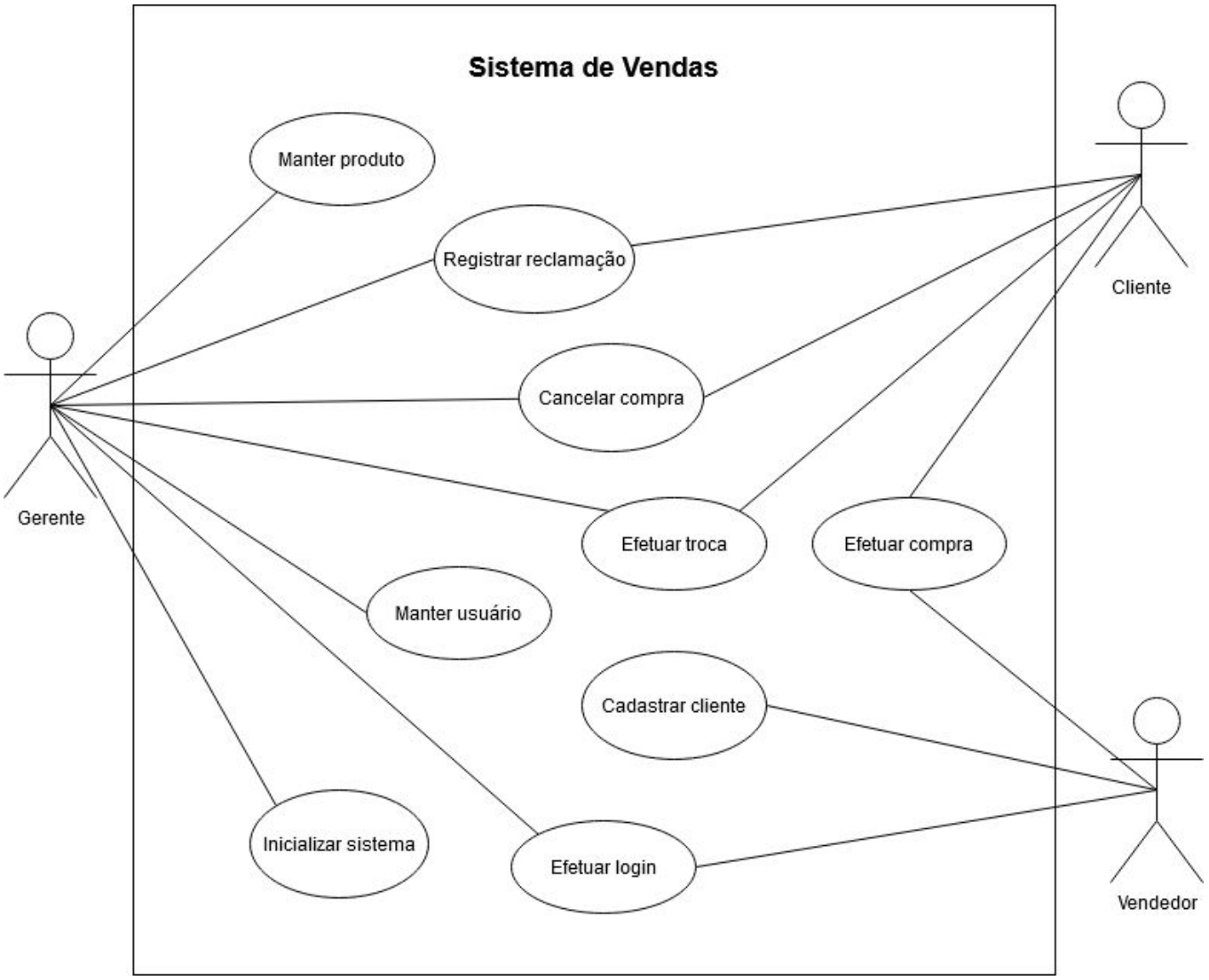
Utilizaremos Java 11 com os frameworks SpringBoot, Hibernate e PostgreSQL.

A aplicação utiliza a biblioteca Jackson JSON, incluída no pacote de dependências Spring Web Starter, para transformar uma instância de POJO em JSON e vice-versa.

2. Representação Arquitetural

Nosso projeto é baseado na arquitetura em camadas, que é o modelo tradicional utilizado no mercado. A divisão em camadas tem como objetivo a possibilidade de se desenvolver o sistema em etapas visando o baixo acoplamento entre as classes, o que torna o software mais fácil de manter caso algum problema seja identificado ou caso seja necessário adicionar alguma outra camada sem que todo o sistema seja comprometido.

3. Visão de Casos de Uso



Apresentamos nesta seção as visualizações dos casos de uso do desenvolvimento do software. Descrevemos aqui os cenários que são focos de interação do sistema. Também procuramos apresentar casos e cenários no qual identificamos funcionalidades relevantes.

Os casos de uso do sistema estão listados abaixo. As descrições dos casos identificados podem ser localizadas posteriormente nessa mesma seção.

Caso de Uso: Efetuar compra

Ator: Vendedor e Cliente

Fluxo normal:

- 1 - Autenticar usuário

- 2 - Vendedor inicia nova venda
- 3 - Vendedor registra itens a serem comprados
- 4 - Sistema calcula total da compra
- 5 - Vendedor registra método de pagamento
- 6 - Cliente efetua o pagamento
- 7 - Sistema gera nota fiscal
- 8 - Sistema registra dados da compra efetuada

Extensões:

- 2a - Se for cliente preferencial, vendedor input o CPF do cliente e cliente confirma seu cadastro
- 4a - Se for cliente preferencial, sistema deve converter total da compra em pontos para compras futuras
- 4aa - Cliente pode solicitar troca de pontos por desconto
- 5a - Se for pagamento em espécie, o sistema deve calcular o troco
- 5b - Se for em cartão, o vendedor passa o cartão no leitor e pede a senha do cliente
- 5c - Se for pix, realiza a leitura do QR Code, insere o valor da transação e pede a senha do cliente
- 6a - Se pagamento falhar, cliente pode escolher outro método de pagamento
- 6b - Se solicitado, gerente pode cancelar a compra

Caso de Uso: Manter Produto

Ator: Gerente

Fluxo normal:

- 1 - Sistema faz autenticação verificando se usuário é válido
- 2 - Gerente realiza CRUD de produto
- 3 - Sistema mantém o estoque dos produtos cadastrados

Extensões:

- 1a - Se usuário for inválido, sistema nega o acesso e pede login novamente
- 2a - Se usuário não for gerente, sistema nega o acesso e pede login novamente

Caso de Uso: Manter Usuário

Ator: Gerente

Fluxo normal:

- 1 - Sistema faz autenticação verificando se usuário é válido
- 2 - Gerente realiza CRUD de usuários

- 3 - Sistema mantém usuários cadastrados pelo gerente

Extensões:

- 1a - Se usuário for inválido, sistema nega o acesso e pede login novamente
- 2a - Se usuário não for gerente, sistema nega o acesso e pede login novamente

Caso de Uso: Efetuar Login

Ator: Gerente ou Vendedor

Fluxo normal:

- 1 - Usuário informa seu login e senha
- 2 - Sistema faz autenticação verificando se o usuário é válido
- 3 - Sistema permite acesso do usuário

Extensões:

- 3a - Se usuário for inválido, sistema nega o acesso e pede login novamente

Caso de Uso: Cadastrar Cliente

Ator: Vendedor

Fluxo normal:

- 1 - Cliente informa os dados necessários.
- 2 - Vendedor preenche os dados do cliente na interface.
- 3 - Vendedor envia os dados ao sistema.

Extensões:

- 2a - Se o cliente for preferencial, o vendedor deve informar ao sistema.

Caso de Uso: Cancelar Compra

Ator: Gerente

Fluxo normal:

- 1 - Vendedor informa o gerente sobre o cancelamento.
- 2 - Gerente autoriza o cancelamento no sistema.
- 3 - Sistema cancela a compra.

Caso de Uso: Efetuar Troca

Ator: Gerente

Fluxo normal:

- 1 - Cliente entrega o produto que será trocado ao gerente.
- 2 - Cliente entrega a nota fiscal da compra ao gerente.
- 3 - Gerente verifica a compra pela nota fiscal no sistema.
- 4 - Gerente gera um vale-troca no mesmo valor do produto no dia da compra.
- 5 - Gerente registra a troca no sistema.

Extensões:

- 3a - Se o gerente não encontrar a compra no sistema, a troca não é realizada.

Caso de Uso: Registrar Reclamação

Ator: Gerente

Fluxo normal:

- 1 - Cliente faz reclamação ao gerente.
- 2 - Gerente busca cliente no sistema por cpf
- 3 - Gerente registra a reclamação no sistema.

Extensões:

- 2a- Se o cliente não estiver registrado o gerente efetua o cadastro

4. Visão Lógica

Apresentamos a descrição da visualização lógica da arquitetura do sistema. Buscamos descrever as classes mais importantes além da sua organização em pacotes. Também descrevemos as realizações de casos de uso mais importantes e aspectos dinâmicos da arquitetura.

Como especificado pela atividade, a implementação da aplicação conta com 6 padrões de projeto para atender às diferentes necessidades do código e boas práticas. A seguir será listado cada padrão pensado pelo grupo e a justificativa para seu uso.

- **MVC**

Para separar as responsabilidades, usamos o padrão Model View Controller. Componentes Model ficam no pacote domain, View é toda a parte do frontend, que apresenta a informação e Controllers que fazem a comunicação entre frontend e

backend ficam no pacote web. Vale ressaltar que este padrão também foi utilizado no frontend.

- **Singleton**

O padrão singleton foi utilizado para a classe MapperUtils. Essa classe é um conversor genérico de uma classe para outra, usado nos controllers para converter entre DTOs e entidades, encapsulando métodos da biblioteca ModelMapper que cuida da lógica da conversão. Como apenas uma instância é necessária para essa classe, o padrão singleton foi escolhido.

- **Builder**

Como uma nota fiscal é formada por muitos campos, foi escolhido padrão builder para construir uma nota fiscal ao finalizar uma venda. O controller recebe o id da venda e o service monta a nota fiscal com os campos da venda encontrada, no método gerar NotaFiscal(venda).

- **Strategy**

Como uma venda pode ter seu pagamento registrado de várias maneiras, o padrão Strategy foi escolhido para implementar as diferentes lógicas possíveis para o pagamento de uma venda. O comportamento para Pagamento é especificado ao se instanciar uma das estratégias: PagamentoDinheiro, PagamentoDebito, PagamentoCredito ou PagamentoPix.

- **DTO**

O padrão DTO (Data Transfer Object) é usado para transferir dados entre as camadas, separando o objeto que lida com o banco, uma entity, do exterior da aplicação. O DTO só tem os campos que serão passados pelo controller para o front é não apresenta lógica de comportamento dentro dele.

- **Facade**

Na camada de serviço, a implementação de toda a lógica fica separada, escondida pela sua interface.

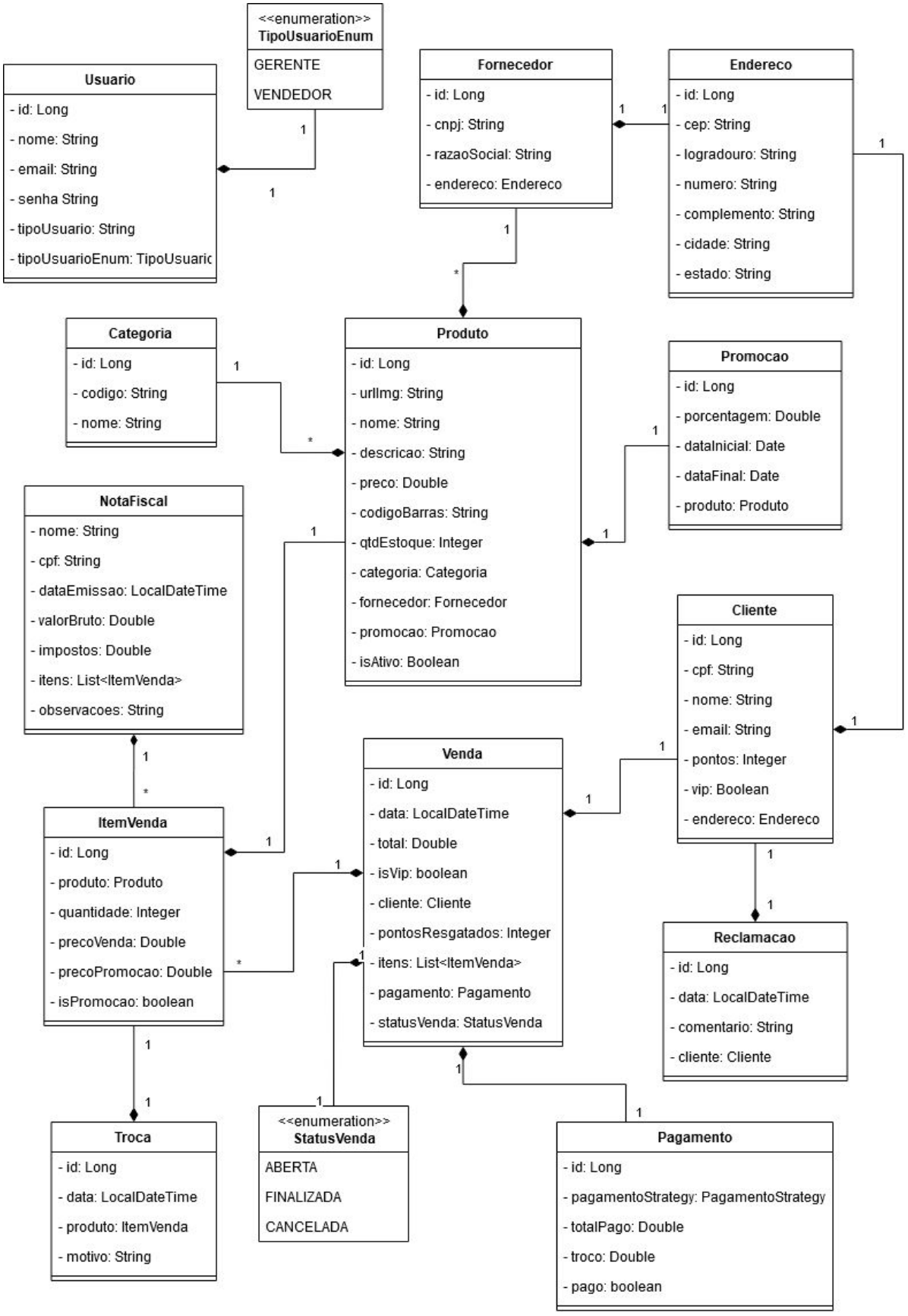
- **Módulo**

O padrão modular foi utilizado no frontend, com o objetivo de dividir a aplicação em pequenos componentes que podem ser reutilizados, tornando o código mais enxuto e de fácil manutenção.

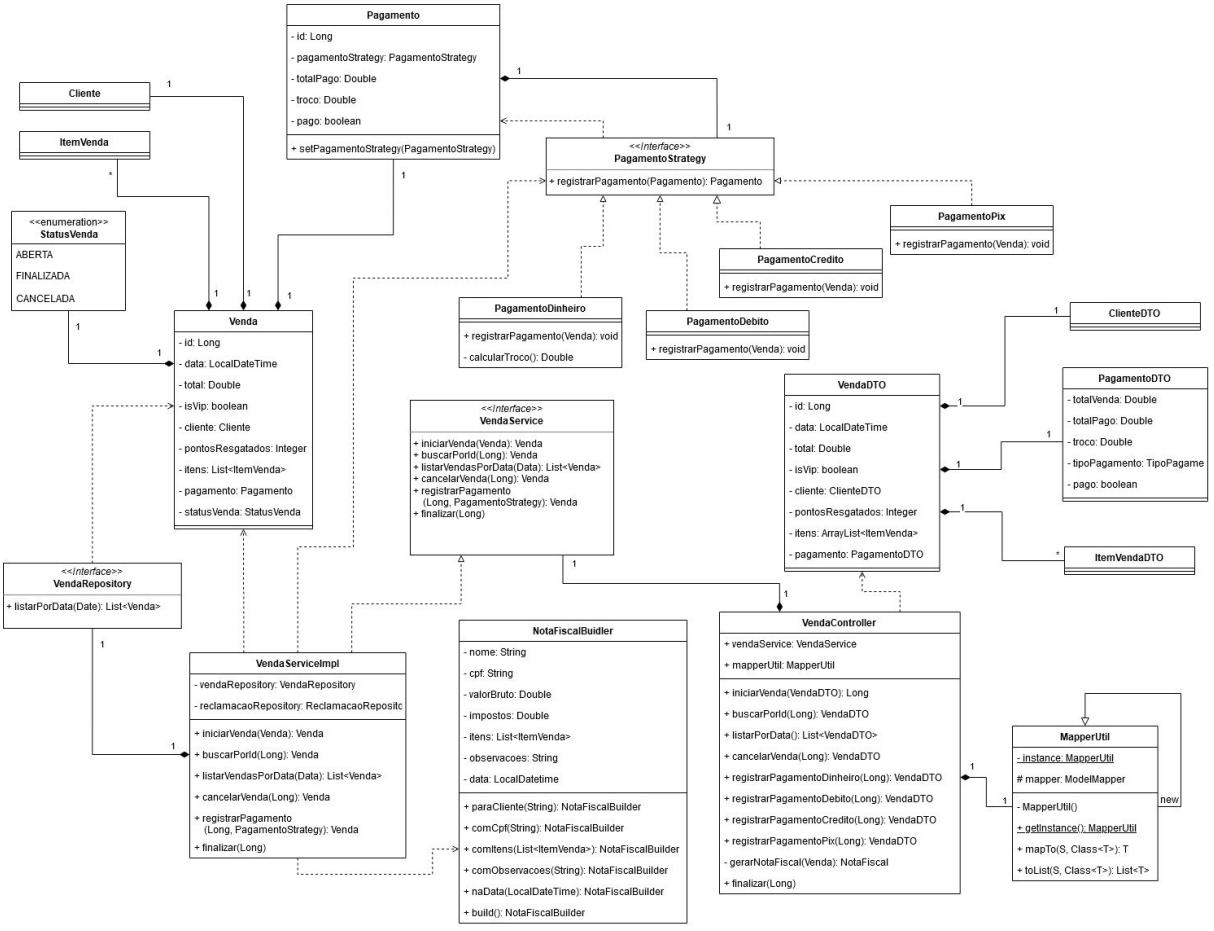
O padrão DAO também é utilizado, mas como já é implementado pelo próprio framework com a interface JpaRepository, o grupo decidiu usar o que o framework oferece.
A seguir apresentamos diagramas de classes para as classes de domínio e para alguns dos casos de uso.

4.1 Diagramas de Classe

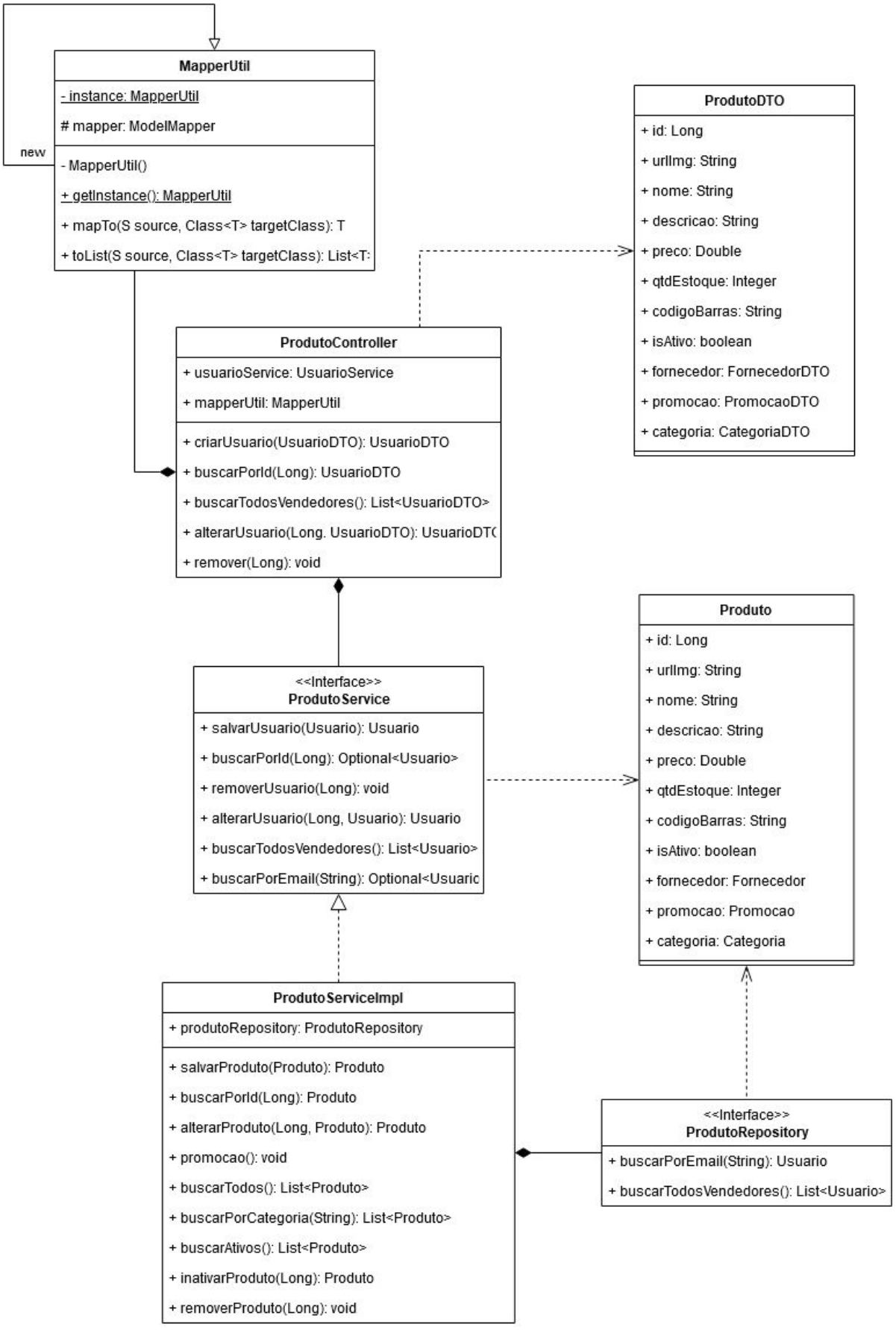
Classes de domínio



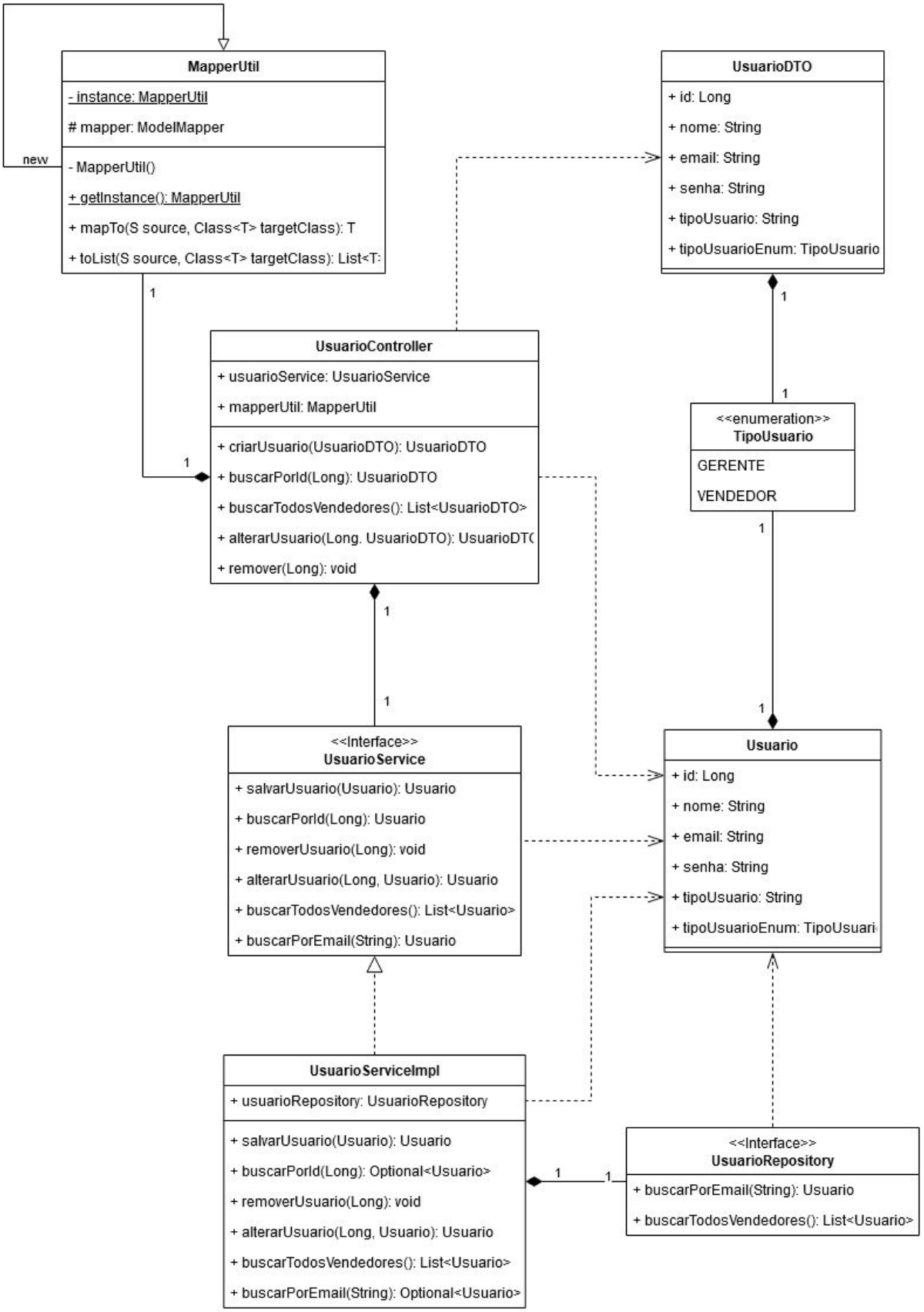
Caso de uso Efetuar Compra



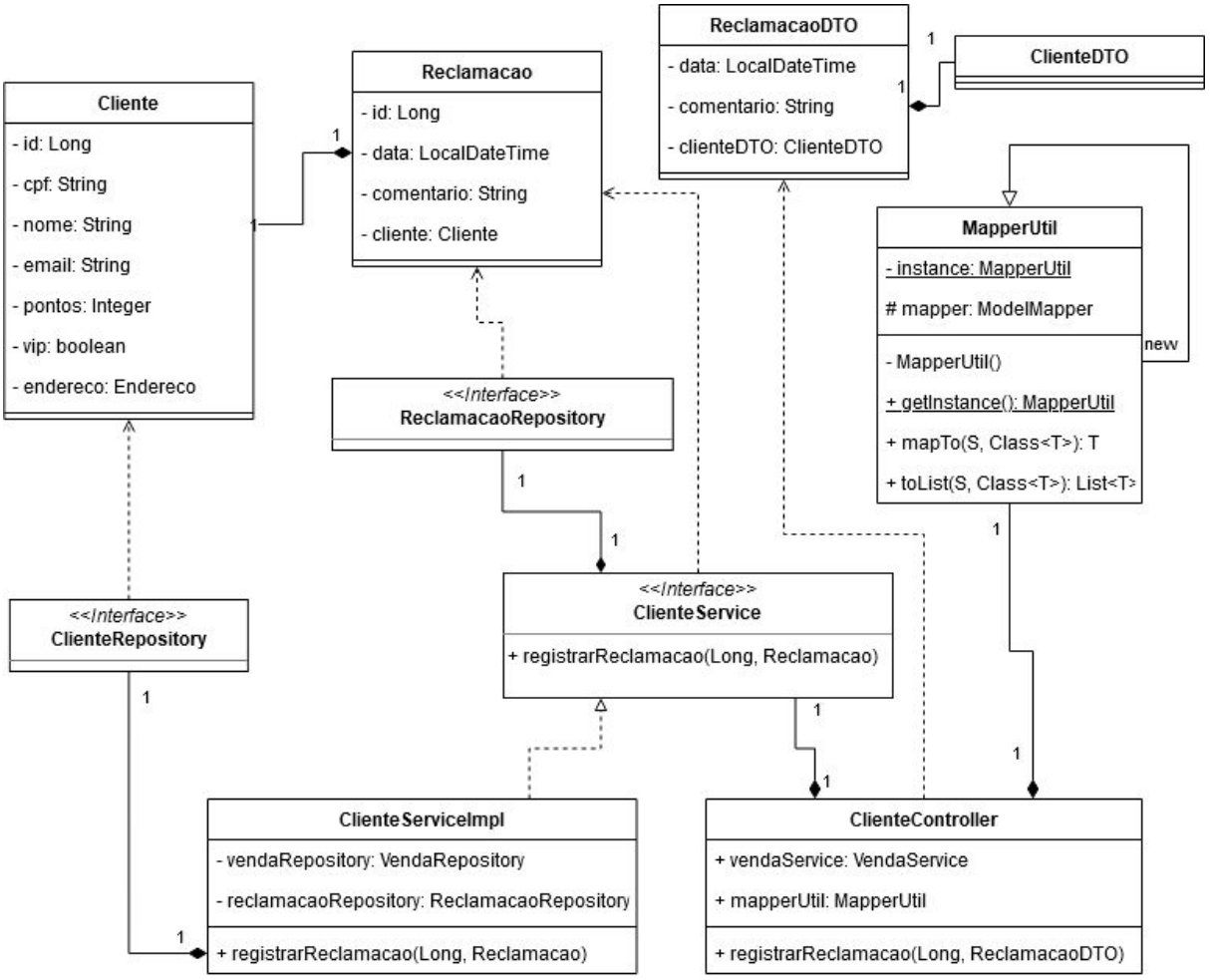
Caso de uso Manter Produto



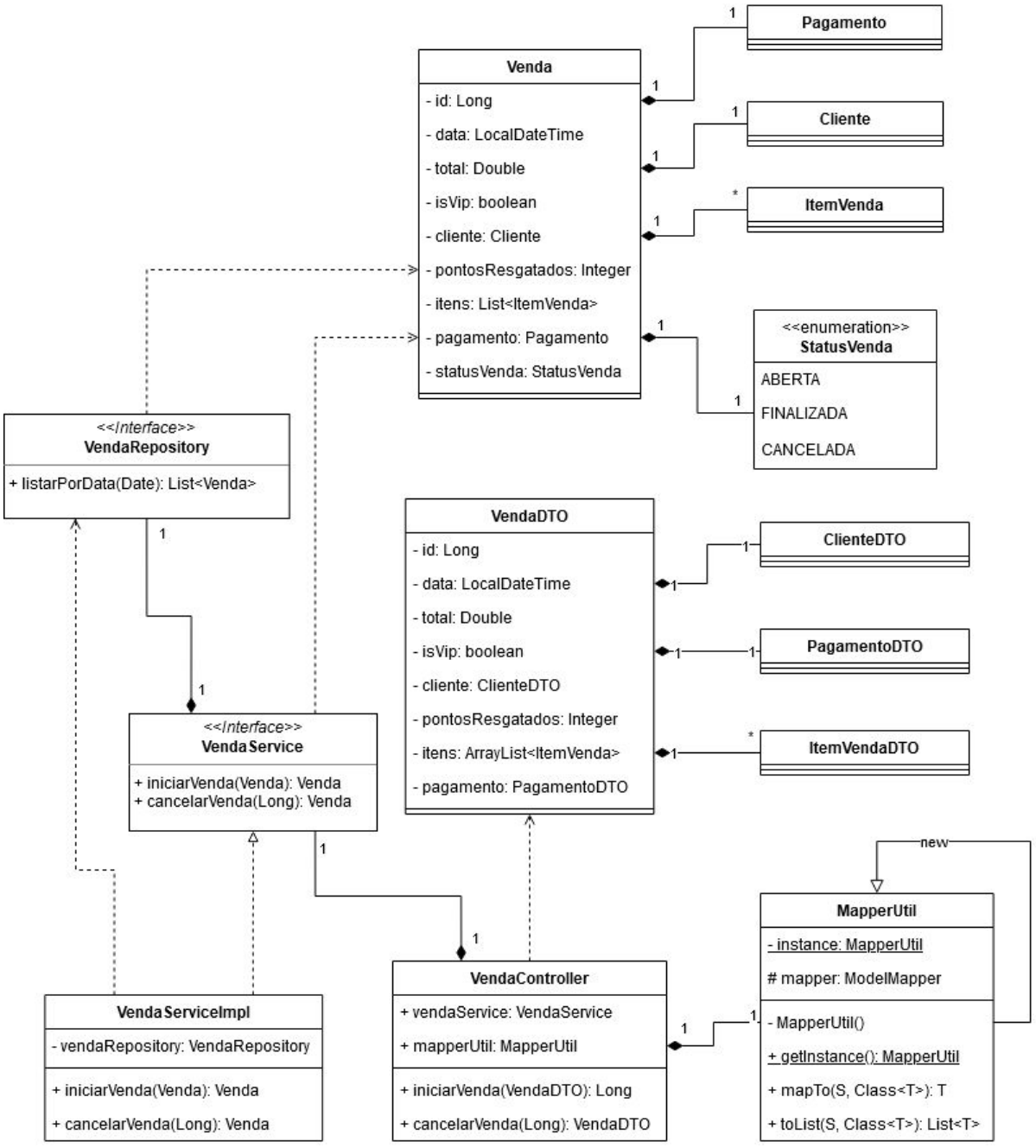
Caso de uso Manter Usuário



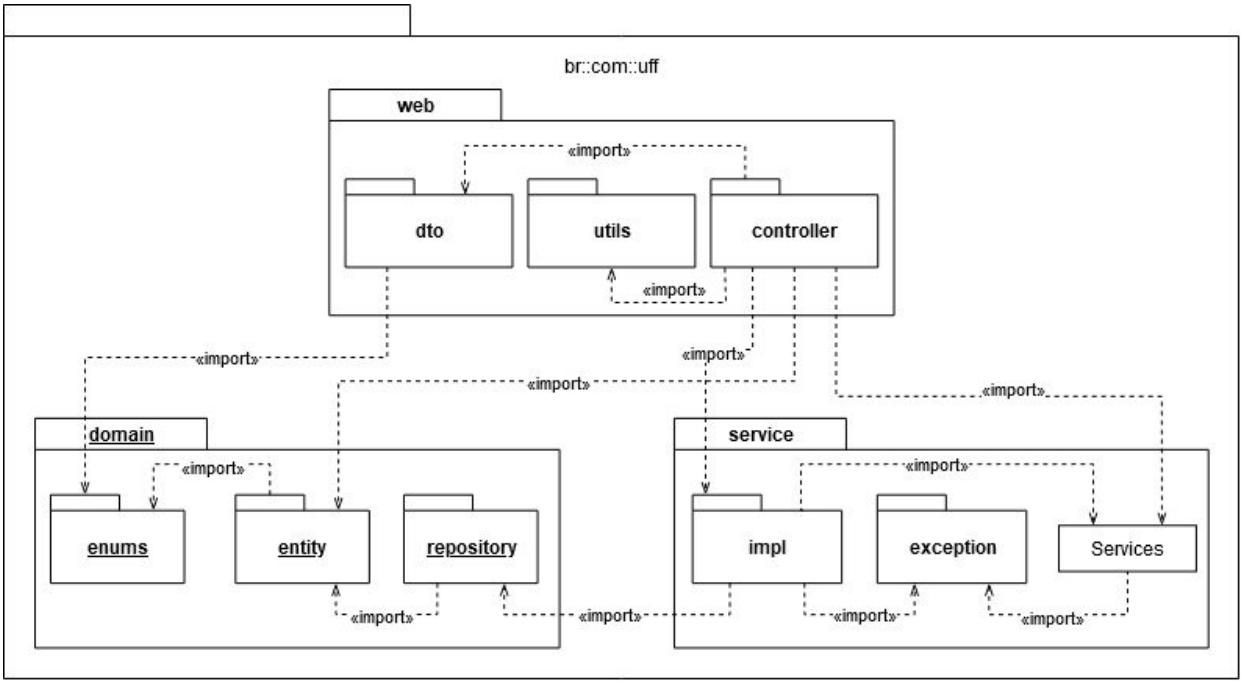
Caso de uso Registrar reclamação



Caso de uso Cancelar compra



4.2 Diagramas de Pacotes

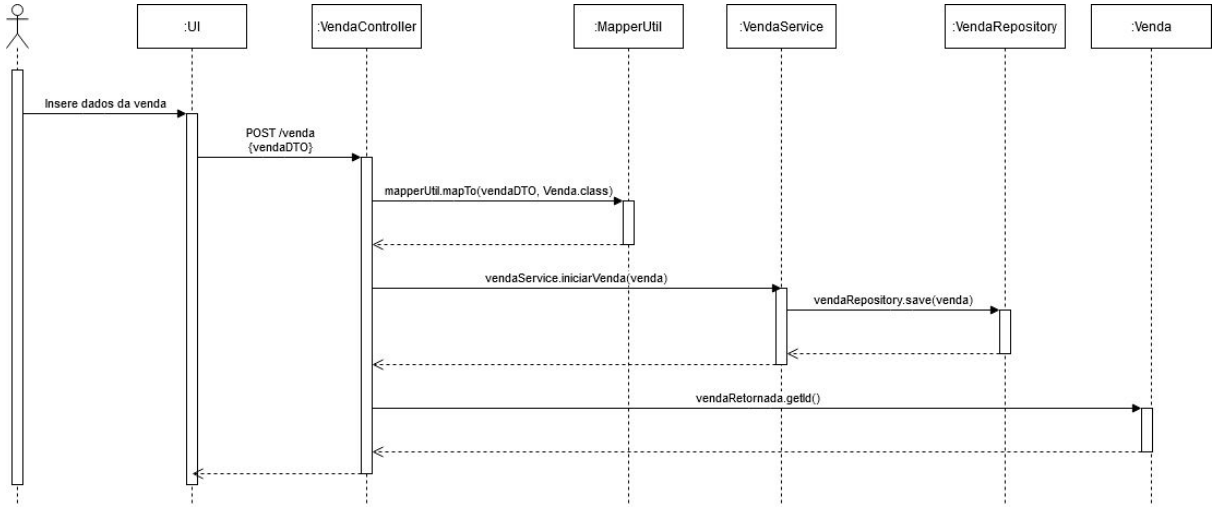


5. Visão de Processos

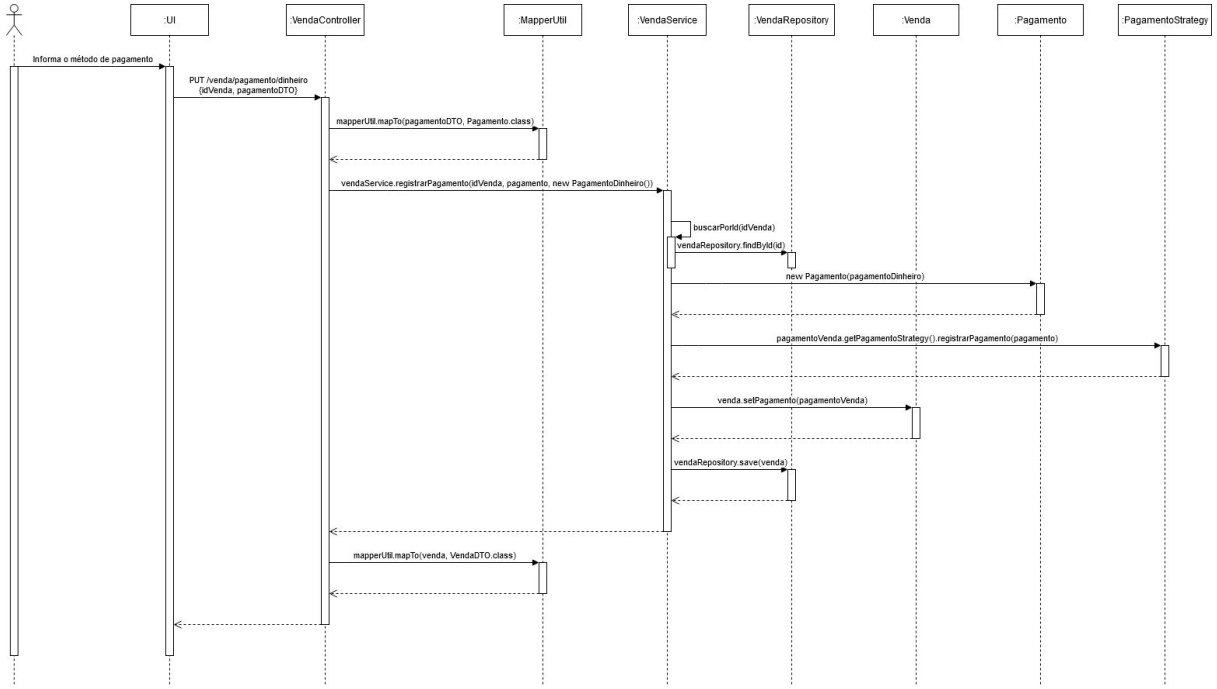
Nesta seção descrevemos a visualização dos processos do negócio relacionados com a arquitetura. Descrevemos as tarefas, processos e encadeamentos envolvidos na execução do sistema e suas interações. Também descrevemos aqui a alocação de objetos e classes para tarefas.

5.1 Diagramas de Sequência

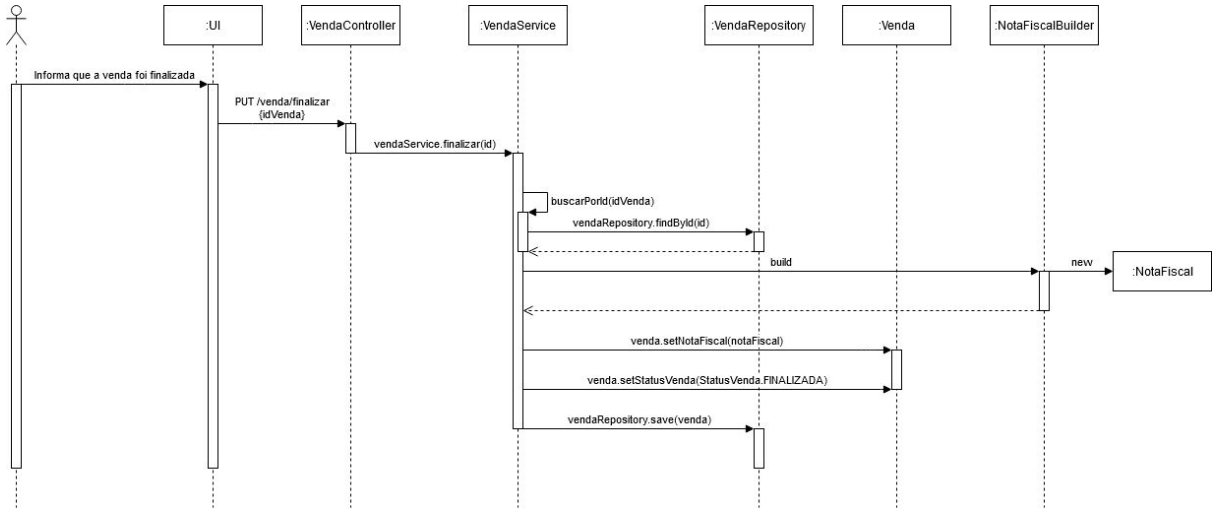
Iniciar venda



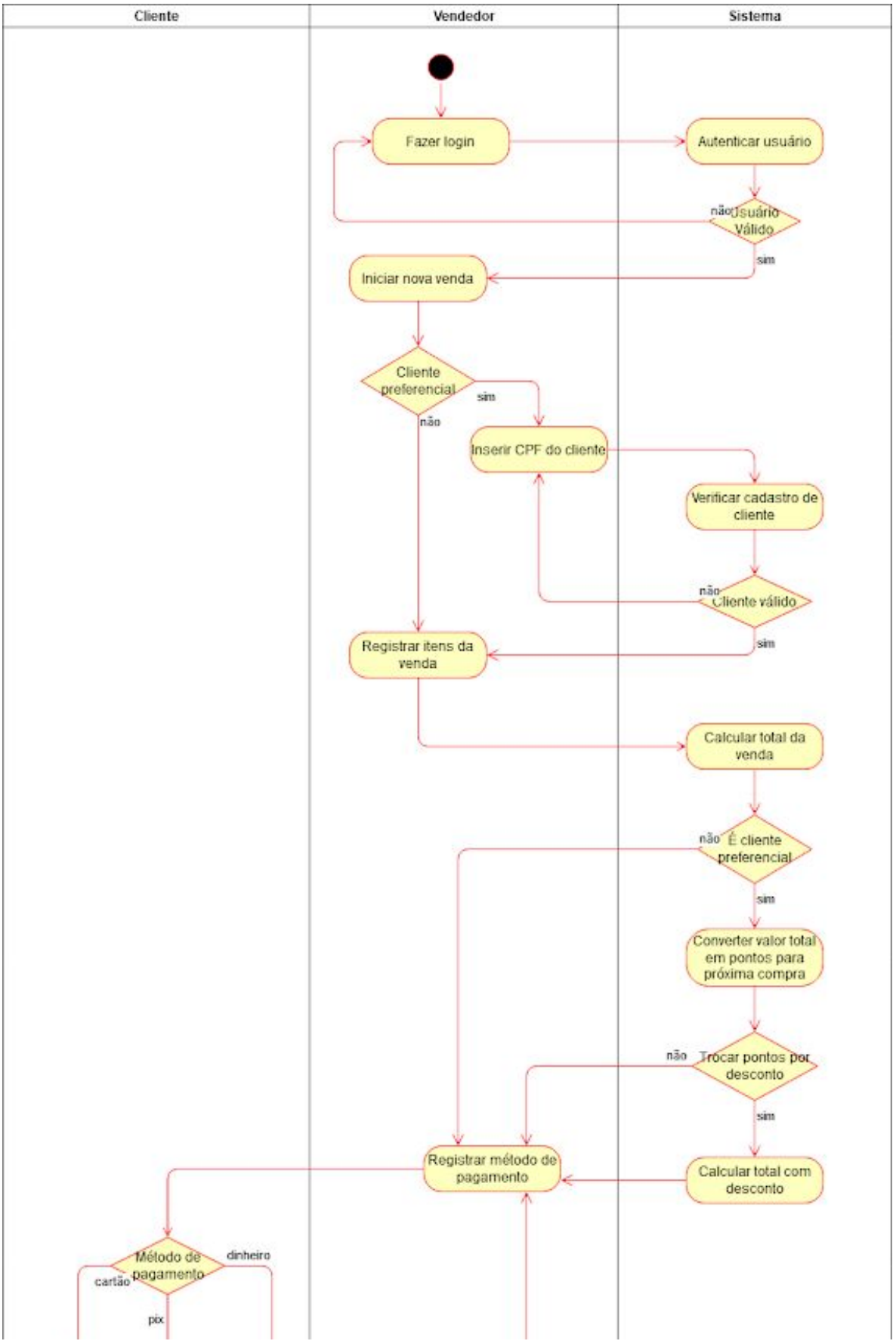
Registrar Pagamento

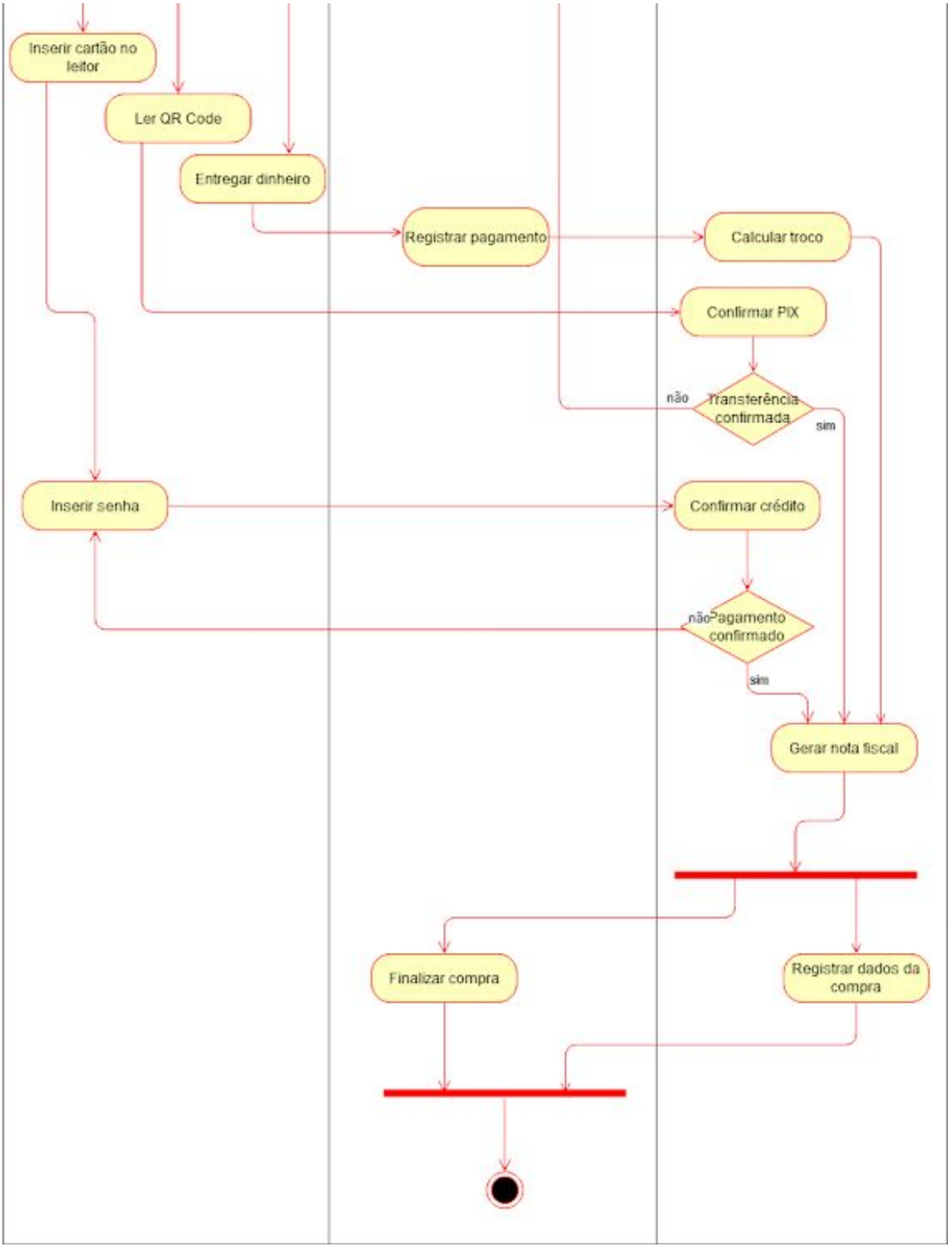


Finalizar venda



5.3 Diagrama de Atividade




Inserir cartão no leitor
Ler QR Code
Entregar dinheiro
Registrar pagamento
Calcular troco
Confirmar PIX
não
Transferência confirmada
sim
Inserir senha
Confirmar crédito
não
Pagamento confirmado
sim
Gerar nota fiscal
Finalizar compra
Registrar dados da compra

6. Visão da Implementação

Citamos aqui a visualização da arquitetura que descreve os diversos componentes físicos para as configurações da plataforma. Também é descrito a alocação de tarefas para os componentes físicos.

No cenário de produção, a aplicação ficará toda na nuvem, usando os serviços da AWS. O servidor de banco de dados PostgreSQL ficará numa instância RDS, o deploy da aplicação será feito em uma instância EC2 rodando CentOS e será executada em um servidor Apache Tomcat. O arquivo estático do frontend será carregado em uma instância S3 para que o dispositivo do usuário acesse a aplicação pelo seu browser.