TRABALHO DA DISCIPLINA DE PROJETO DE SOFTWARE

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Grupo:

Breno Alves Fróes Peres - 217083110

Caio Cesar Guimarães Costa - 114031095

Christopher Albino Corrêa - 217083108

Gleison Lima de Souza - 217083117

Milena Crivella Veríssimo da Fonseca - 217083093

Rodolfo Bandeira de Melo - 217083107

Victor Matheus Pereira de Azevedo - 217083124

Victor Rodrigues Marques -217083102

REPOSITÓRIO DA IMPLEMENTAÇÃO

https://github.com/Victormatheus819/TrabalhoPSoft

SUMÁRIO

obs: Para melhor visualização dos diagramas, basta clicar sobre os mesmos que você será redirecionado para visualização no Google Drive.

1 INTRODUÇÃO

1.1 FINALIDADE

Este documento relata detalhadamente a visão arquitetural do sistema de vendas na web, utilizando algumas visões de arquitetura a fim de demonstrar diferentes perspectivas do sistema, além de explanar sobre o projeto e sua concepção. Sendo o intuito majoritário representar as decisões arquiteturais mais relevantes no decorrer do desenvolvimento do sistema.

1.2 ESCOPO

Este documento de Arquitetura de Software provê uma visão geral da arquitetura do sistema de vendas na web.

2 CONCEPÇÃO

2.1 TÍTULO DO DESAFIO OU PROBLEMA

Sistema para realizar vendas com acesso via web.

2.2 MOTIVAÇÃO

Devido ao avanço tecnológico, os sistemas de vendas precisam de atualização assim como as demais tecnologias. Para suprir essa necessidade de inovação um sistema web com autenticação, com ferramentas convencionais de uma aplicação offline e ainda com funcionalidades extras e modernas é bem oportuno.

2.3 OBJETIVO

A proposta é a criação de um site de vendas, que possui cadastros de funcionários para ter acesso ao sistema. O mesmo é inicializado pelos gerentes, mas utilizado por vendedores. Além disso, o sistema possui funções extras como trocas, cancelamentos, mecanismo de pontos e promoções, diferenciação de clientes, etc.

2.4 STAKEHOLDERS

Nome[(1)]	Classificação[(2)]	Interação[(3)]
Gerentes	usuários finais	direta
Vendedores	usuários finais	direta
Donos de comércio	investidores	indireta
Clientes preferenciais	clientes internos	indireta

Clientes convencionais	clientes internos	indireta
PROCON	órgão governamental	indireta

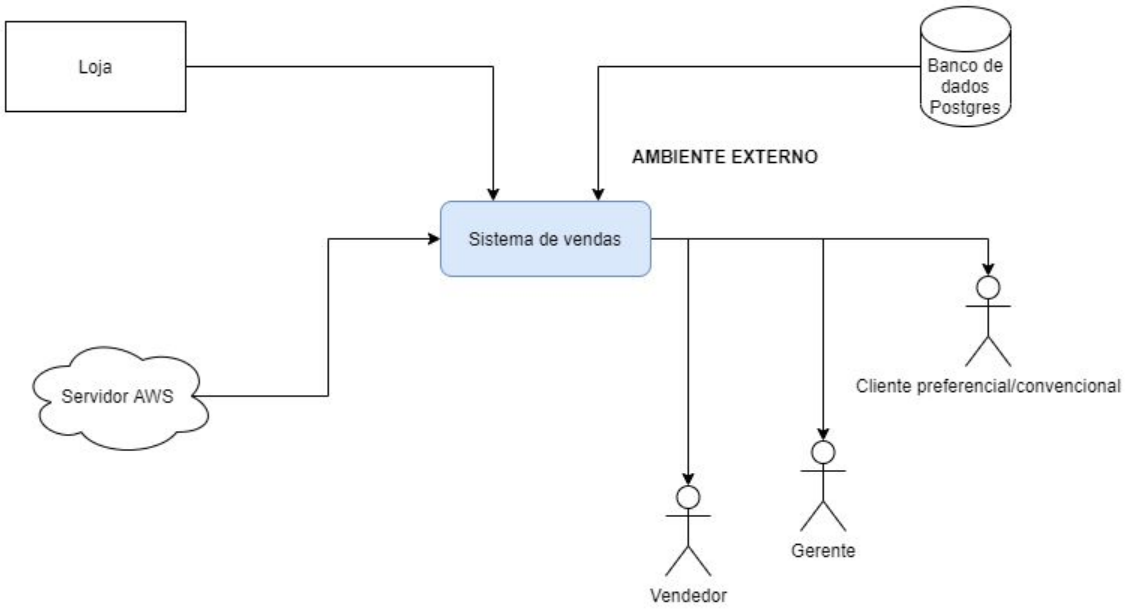
2.5 ANÁLISE DE VIABILIDADE

O sistema pode ser implementado na maioria das linguagens para plataformas web, como Java, além de ser possível disponibilizá-lo em outros tipos de plataformas, como mobile. Sendo um sistema web de implementação relativamente simples, o prazo (caso exista) seria cumprido.

O serviço atual se encontra disponível no estabelecimentos comerciais, porém ainda com bastante limitação. Portanto as vendas atuais são realizadas por vendedores somente em máquinas com o sistema previamente instalado e pronto para uso, o que gera alta dependência desses equipamentos e baixíssimas flexibilidade e portabilidade de uso, a comprometer significativamente a eficiência e eficácia no fluxo de vendas do comércio em questão.

2.6 MODELO DE CONTEXTO

O sistema estaria hospedado em qualquer plataforma robusta que promovesse uma alta disponibilidade do sistema, como AWS. Além disso, deve estar acessível a web em geral, especialmente os stakeholders.



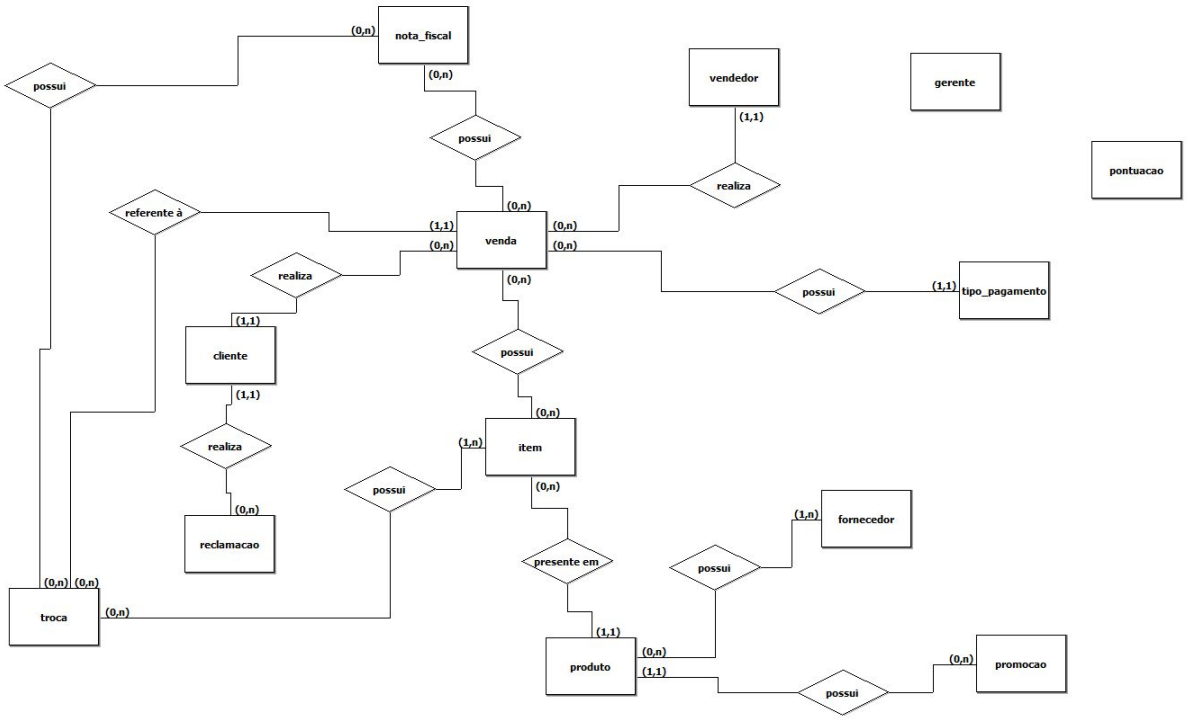
2.7 REQUISITOS

ID [1]	Descrição [2]	Referência [4]
RF-01	O sistema deve registrar a venda de diferentes produtos em uma transação de compra.	
RF-02	O sistema deve calcular o total de produtos comprados em uma transação.	RF-01
RF-03	O sistema deve receber o pagamento que pode ser em espécie, usando pix, cartão de crédito ou débito.	RF-02
RF-04	O sistema deve calcular o troco para pagamentos feitos em espécie.	RF-03
RF-05	Após efetuada a compra, o sistema deve gerar a nota fiscal.	RF-04
RF-06	O sistema deve armazenar todos os dados das compras efetuadas: CPF, identidade, endereço e e-mail.	RF-05
RF-07	O sistema deve ser capaz de identificar clientes preferenciais, através de seu CPF, antes de efetuarem a compra.	
RF-08	O sistema deve permitir que os clientes preferenciais confirmem sua identificação antes da compra	RF-07
RF-09	O sistema deve armazenar o valor total comprado por um cliente preferencial e converter o total em pontos que podem ser utilizados em compras futuras.	RF-07, RF-02
RF-10	O sistema deve permitir que um cliente preferencial troque seu total de pontos por um desconto, durante uma compra.	RF-07, RF-09
RF-11	O sistema deve manter um controle de estoque dos produtos à venda.	RF-01
RF-12	Os produtos devem ser mantidos em um catálogo de produtos contendo o nome, o código de identificação, numeração do código de barras e fornecedor	RF-01
RF-13	O sistema deve manter um cadastro de usuários ativos do sistema que incluem os gerentes e os vendedores (caixas).	
RF-14	Somente usuários autorizados devem acessar o sistema.	RF-13
RF-15	O sistema deve ser inicializado pelo gerente.	
RF-16	Alterações no cadastro de produtos deve ser feito pelo gerente.	RF-13

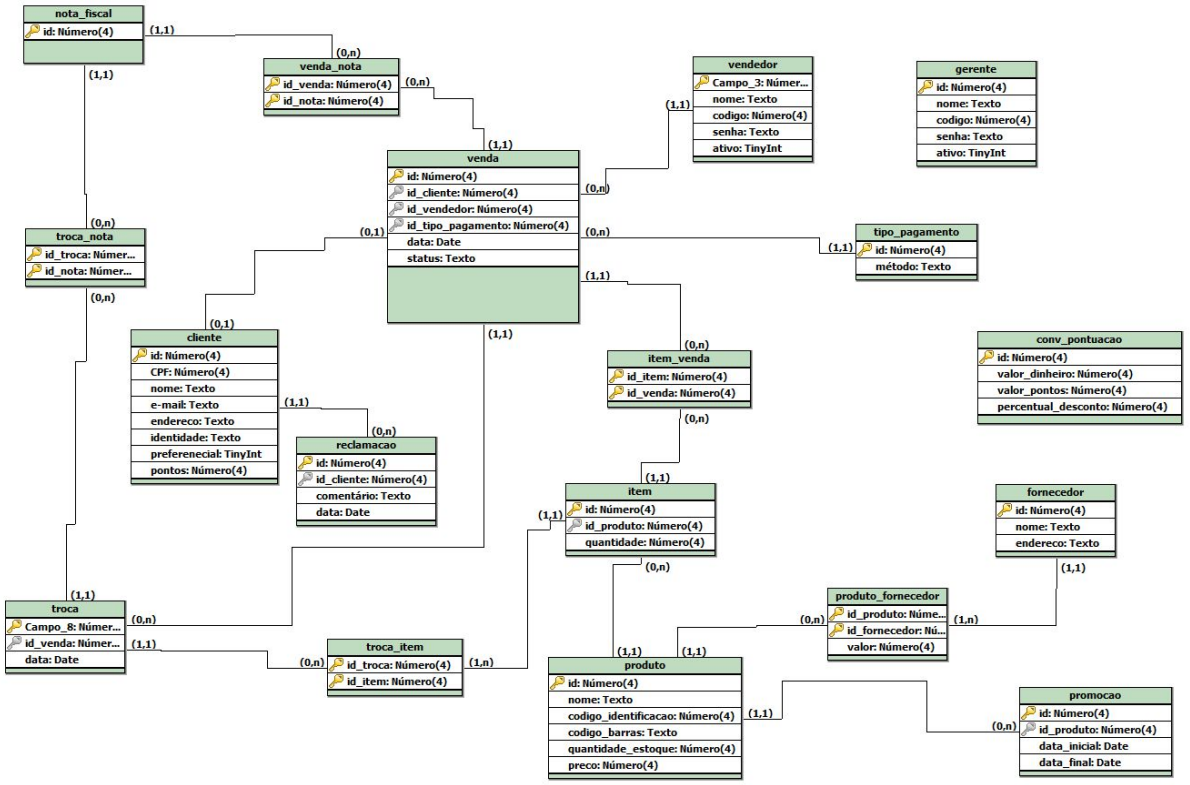
RF-17	O sistema deve permitir que o gerente cancele uma compra.	RF-13
RF-18	Antes de iniciar suas atividades, o vendedor deve se logar no sistema.	RF-13, RF-15
RF-19	A cada 2 meses o sistema deve enviar informações de promoções aos clientes preferenciais.	
RF-20	O sistema deve registrar trocas de produtos feitas por clientes preferenciais ou não	
RF-21	O sistema deve registrar reclamações feitas pelos clientes quer sejam preferenciais ou não em uma data específica.	
RNF-01	O sistema deve ser acessado via Web	
RNF-02	O sistema deve ser implementado em Java	

3 MODELAGEM DO BANCO DE DADOS

3.1 MODELO CONCEITUAL



3.2 MODELO LÓGICO



4. VISÃO DE CASOS DE USO

4.1 CENÁRIO TÍPICO

Gerente acessa o sistema de vendas na web, realiza o seu login e inicializa o sistema para vendas. Vendedor realiza login e inicia uma nova venda já com o sistema iniciado. Após isso, insere o CPF do cliente com o sistema verificando-o se trata-se de um cliente preferencial. Então o vendedor começa a inserir os produtos da compra e simultaneamente o sistema verifica o estoque dos mesmos no sistema da loja e exibe o subtotal da compra. Ao final, vendedor encerra a inserção de novos produtos e solicita ao cliente a forma de pagamento. Então, o sistema verifica e valida a forma de pagamento inserida. Caso esteja tudo certo, o sistema armazena os dados da compra e do cliente, emite a nota fiscal correspondente e então o vendedor finaliza a compra.

4.2 CASOS DE USO

Nome[(1)]	Cenário[(2)]	Ator(es)[(3)]	Referência(s)[(4)]
Logar no sistema	**Pré-condição:** Gerente disponível. **Fluxo normal:** 1. Gerente acessa o sistema web e insere seus dados. 2. Sistema verifica os dados inseridos. 3. Sistema verifica se o usuário está autorizado e ativo no sistema. 4. Sistema fornece as opções para o usuário logado. **Fluxo alternativo:** 2.1 Usuário não encontrado. 2.1.1 Sistema informa erro na inserção dos dados; 2.1.2 Retorna ao passo 1; 3.1 Usuário inativo/desautorizado. 3.1.1 Sistema informa ao usuário a restrição quanto ao seu login; 3.1.2 Sistema informa erro de autorização/atividade; 3.1.3 Retorna ao passo 1.	Gerente/ Sistema	RF-13, RF-14
Iniciar sistema para vendas	**Pré-condição:** Gerente logado no sistema. **Fluxo normal:** 1. Gerente habilita o sistema para vendas.	Gerente/ Vendedor /Sistema	RF-13, RF-14, RF-15, RF-18

	2. Vendedor acessa o sistema web e insere seus dados. 3. Sistema verifica os dados inseridos. 4. Sistema verifica se o usuário está autorizado e ativo no sistema. 5. Sistema é iniciado e fica disponível para operações. **Fluxo alternativo:** 3.1 Usuário não encontrado. 3.1.1 Sistema informa erro na inserção dos dados; 3.1.2 Retorna ao passo 2; 4.1 Usuário inativo/desautorizado. 4.1.1 Sistema informa ao usuário a restrição quanto ao seu login; 4.1.2 Sistema informa erro de autorização/atividade; 4.1.3 Retorna ao passo 2.		
Cadastrar usuário	**Pré-condição:** Gerente logado no sistema. **Fluxo normal:** 1. Gerente escolhe a opção de cadastro; 2. Sistema exibe um formulário de cadastro para o usuário; 3. Gerente insere os dados pertinentes ao formulário; 4. Sistema verifica corretude e preenchimento dos dados; 5. Sistema informa sucesso no cadastro realizado pelo usuário; 6. Sistema redireciona usuário para a página inicial. **Fluxo alternativo:** 4.1 Erro na entrada dos dados do usuário	Vendedor /Gerente/ Sistema	RF-13, RF-18

	4.1.1 Sistema informa o erro pertinente na inserção dos dados; 4.1.2 Retorna ao passo 3.		
Efetuar compra	**Pré-condição:** Sistema inicializado e usuário com cadastro realizado. **Fluxo normal:** 1. Vendedor começa uma nova venda. 2. Vendedor insere o CPF do cliente. 3. Sistema verifica se o cliente é preferencial. 4. Vendedor insere os produtos da compra em questão. 5. Sistema verifica o estoque de cada produto. (paralelo) 6. Sistema exibe subtotal da compra. (paralelo) 7. Sistema mostra o total da compra. 8. Vendedor insere a forma de pagamento (PIX, espécie, cartão de débito, cartão de crédito) escolhida pelo cliente. 9. Sistema valida forma de pagamento. 10. Sistema armazena os dados da compra e do cliente. 11. Sistema emite a nota fiscal da compra. 12. Vendedor finaliza a compra. **Fluxo alternativo:** 5.1 Produto fora de estoque. 5.1.1 Sistema mostra erro de que produto está fora de estoque; 5.1.2 Vendedor informa ao cliente a indisponibilidade do produto; 5.1.3 Retorna ao passo 4.	Gerente/ Vendedor /Sistema	RF-01, RF-02, RF-03, RF-04, RF-05, RF-06, RF-07, RF-08, RF-09, RF-10, RF-11, RF-12

	7.1 Cliente preferencial utiliza pontos. 7.1.1 Vendedor insere os pontos do cliente; 7.1.2 Sistema converte os pontos em desconto; 7.1.3 Vendedor informa o novo total da compra; 7.1.4 Retorna ao passo 8. 7.2 Compra cancelada 7.2.1 Vendedor informa ao gerente sobre cancelamento da compra; 7.2.2 Gerente autoriza o cancelamento através de uma senha; 7.2.3 Retorna ao passo 1. 8.1 Pagamento em espécie 8.1.1 Vendedor insere quanto em espécie o cliente pagou; 8.1.2 Sistema calcula o troco e o informa ao cliente; 8.1.3 Retorna ao passo 9. 9.1 Pagamento não aprovado 9.1.1 Sistema verifica algum problema com o cartão de crédito/débito ou com o PIX; 9.1.2 Vendedor informa ao cliente a falha na realização da compra; 9.1.3 Retorna ao passo 8. 10.1 Compra de um cliente preferencial. 10.1.1 Sistema converte o total da compra em pontos; 9.2.2 Retorna ao passo 10.		

Realizar troca de produto	**Pré-condição:** Compra realizada com sucesso e cliente com a nota fiscal correspondente. Gerente logado no sistema. 1. Gerente seleciona a opção troca. 2. Gerente insere informações da compra constatadas na NF. 3. Sistema busca compra correspondente. 4. Gerente seleciona os produtos da compra a serem trocados. 5. Sistema recupera valor da compra original. 6. Gerente insere produtos de mesmo valor. 7. Sistema verifica o estoque de cada produto. 8. Sistema calcula valor da nova compra. 9. Sistema registra troca dos produtos. 10. Gerente finaliza a troca. 11. Gerente desloga do sistema. 12. Sistema finaliza sessão. 13. Sistema redireciona usuário para a página inicial. **Fluxo alternativo:** 7.1 Produto fora de estoque. 7.1.1 Sistema mostra erro de que produto está fora de estoque; 7.1.2 Vendedor informa ao cliente a indisponibilidade do produto; 7.1.3 Retorna ao passo 6. 8.1 Valor da nova compra diferente da compra original. 8.1.1 Sistema mostra a diferença de valor entre a nova compra e a compra original	Gerente/ Sistema	RF-13, RF-14, RF-15, RF-20

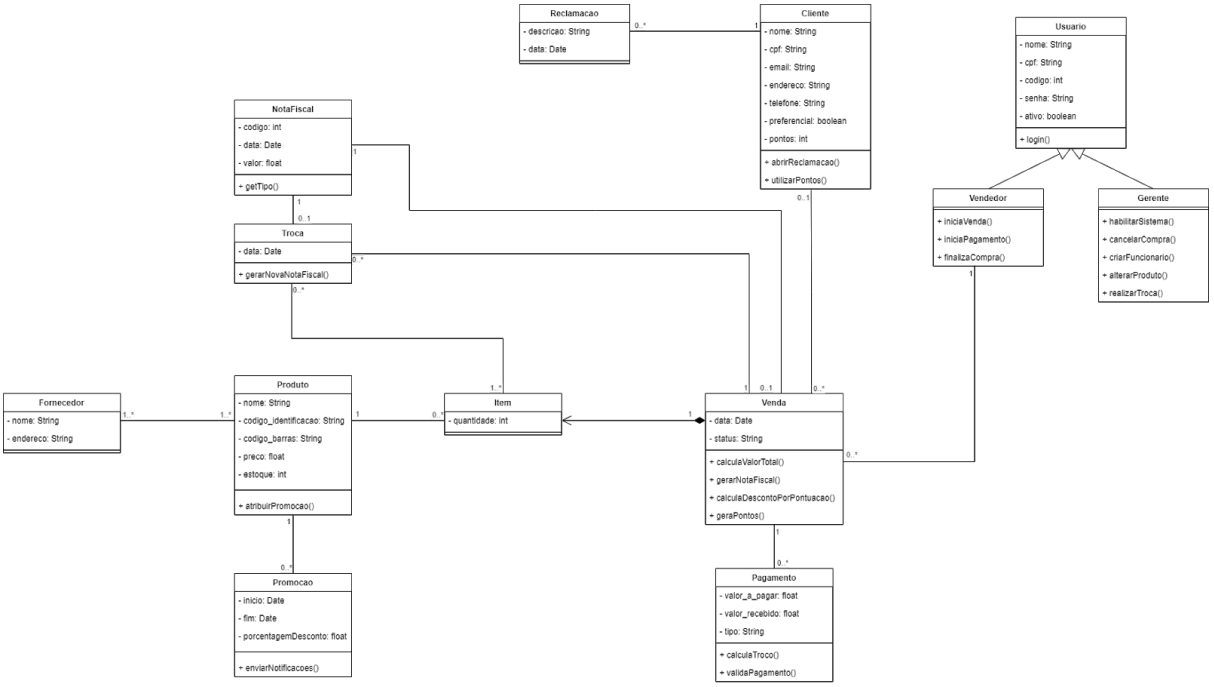
	8.1.2 Vendedor insere a forma de pagamento (PIX, espécie, cartão de débito, cartão de crédito) escolhida pelo cliente. 8.1.3 Sistema valida a forma de pagamento. 8.1.4 Emitir nova nota fiscal. 8.1.5 Retorna ao passo 9. 8.2 Pagamento em espécie 8.2.1.1 Vendedor insere quanto em espécie o cliente pagou; 8.2.1.2 Sistema calcula o troco e o informa ao cliente; 8.2.1.3 Retorna ao passo 8.1.3. 8.3 Pagamento não aprovado 8.3.1.1 Sistema verifica algum problema com o cartão de crédito/débito ou com o PIX; 8.3.1.2 Vendedor informa ao cliente a falha na realização da compra; 8.3.1.3 Retorna ao passo 8.1.2.		
Alterar informações de produto	**Pré-condição:** Produto(s) cadastrado(s) na plataforma. Gerente logado no sistema. **Fluxo normal:** 1. Gerente acessa o catálogo de produtos. 2. Gerente seleciona a opção de editar o produto determinado. 3. Gerente insere as informações a serem atualizadas. 4. Gerente revisa as informações. 5. Finalizar tarefa. 6. Gerente desloga do sistema.. 7. Sistema finaliza sessão.	Gerente/ Sistema	RF-12, RF-13, RF-14, RF-15, RF-16

	8. Sistema redireciona usuário para a página inicial.		

4.3 DIAGRAMA DE CASO DE USO

5 VISÃO LÓGICA

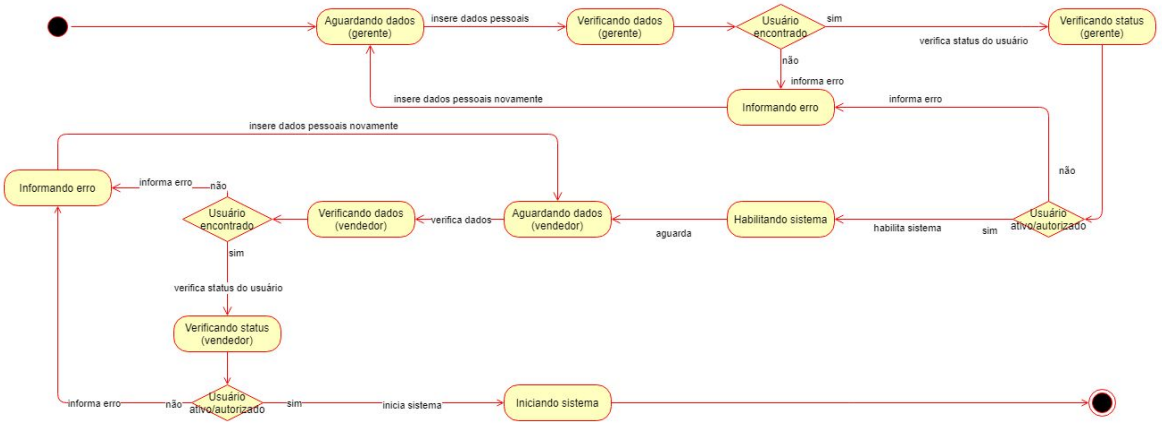
5.1 DIAGRAMA DE CLASSES



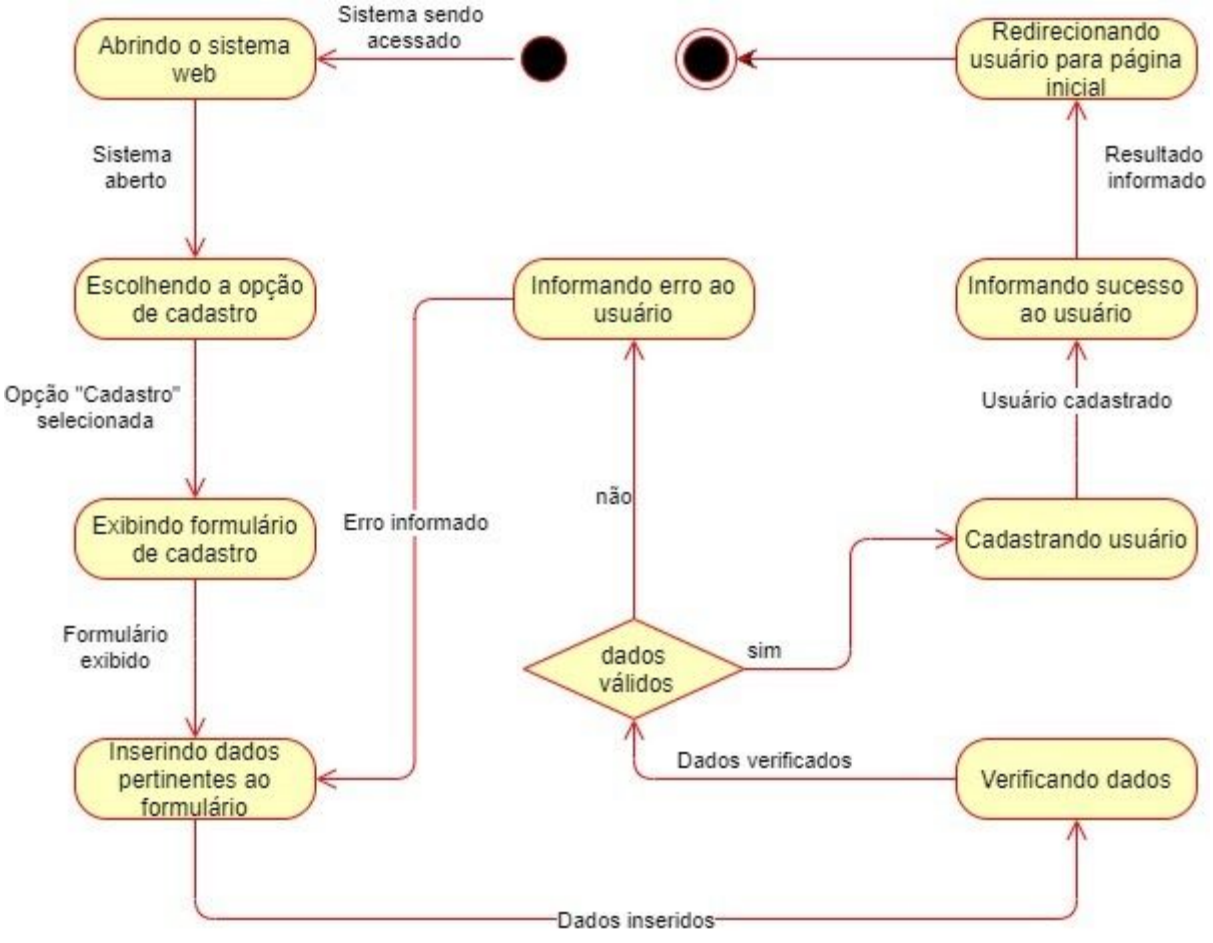
5.2 DIAGRAMA DE ESTADO

5.2.1 LOGAR NO SISTEMA

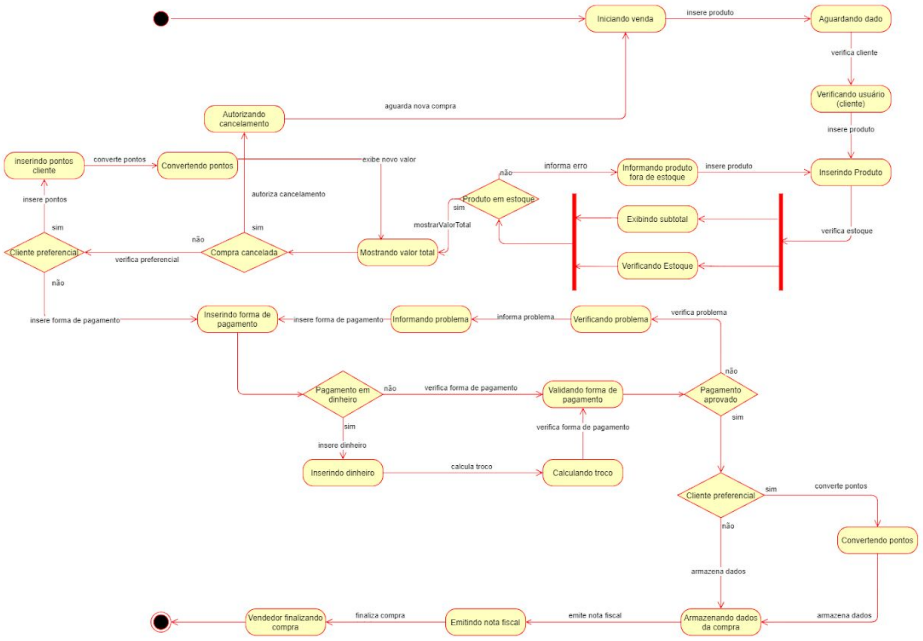
5.2.2 INICIAR SISTEMA



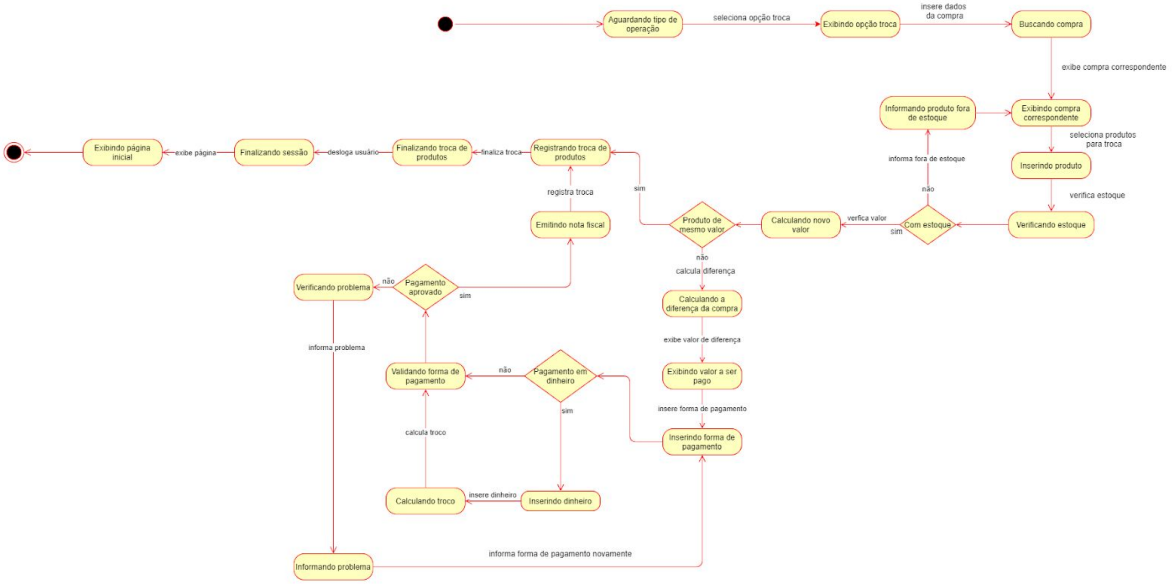
5.2.3 CADASTRAR USUÁRIO



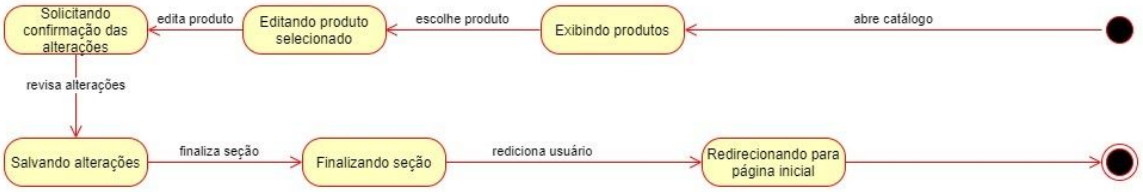
5.2.4 EFETUAR COMPRA



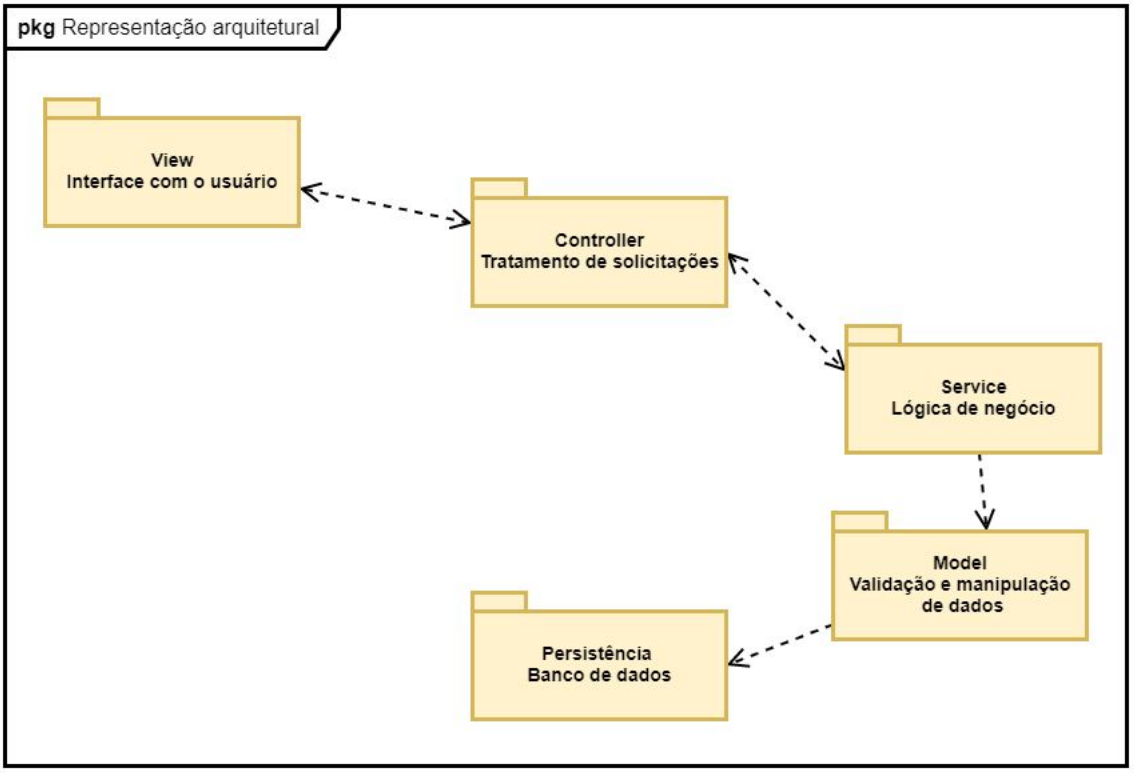
5.2.5 TROCAR PRODUTO



5.2.6 ALTERAR INFORMAÇÕES DE PRODUTOS

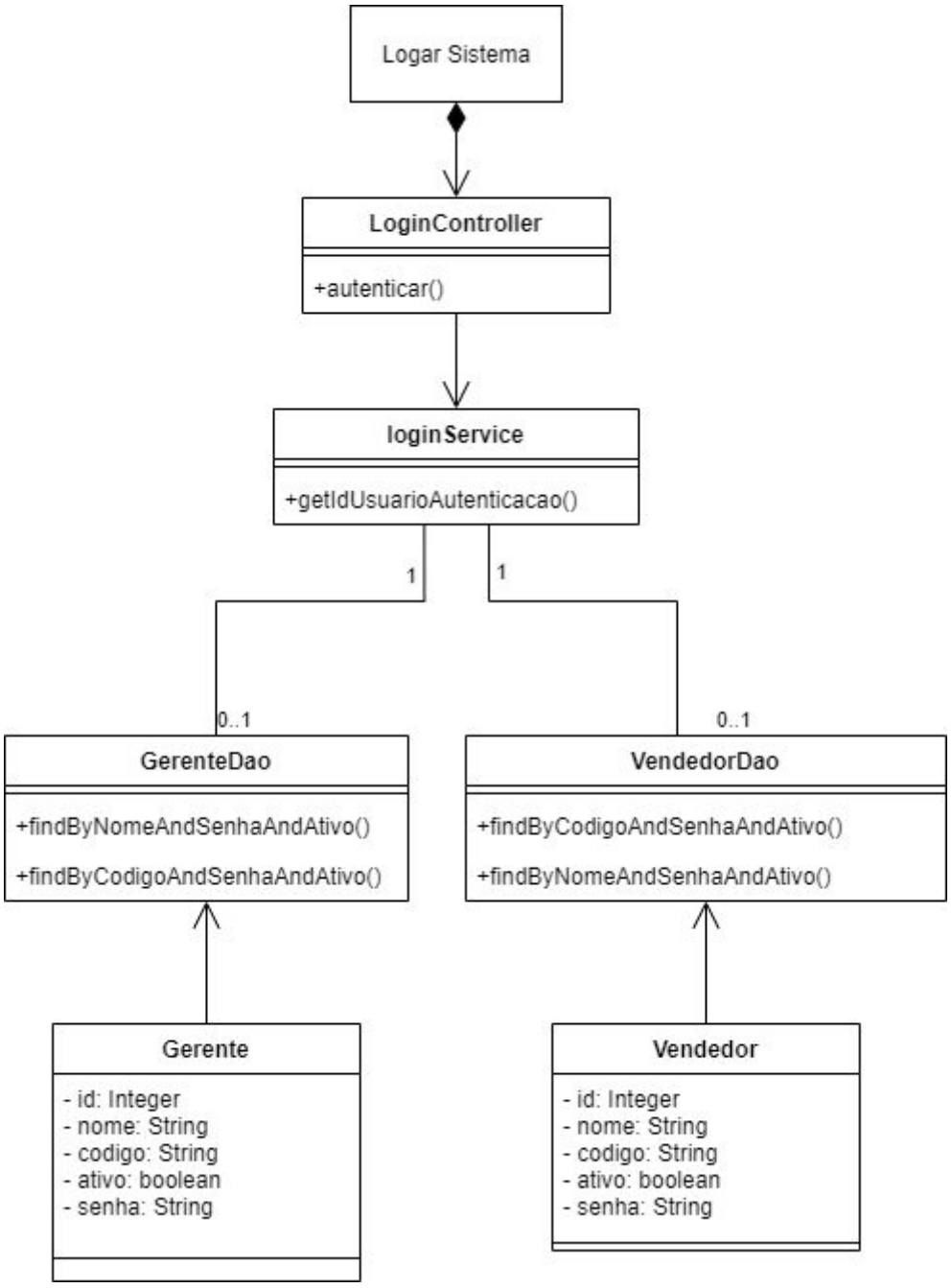


5.3 DIAGRAMA DE REPRESENTAÇÃO ARQUITETURAL

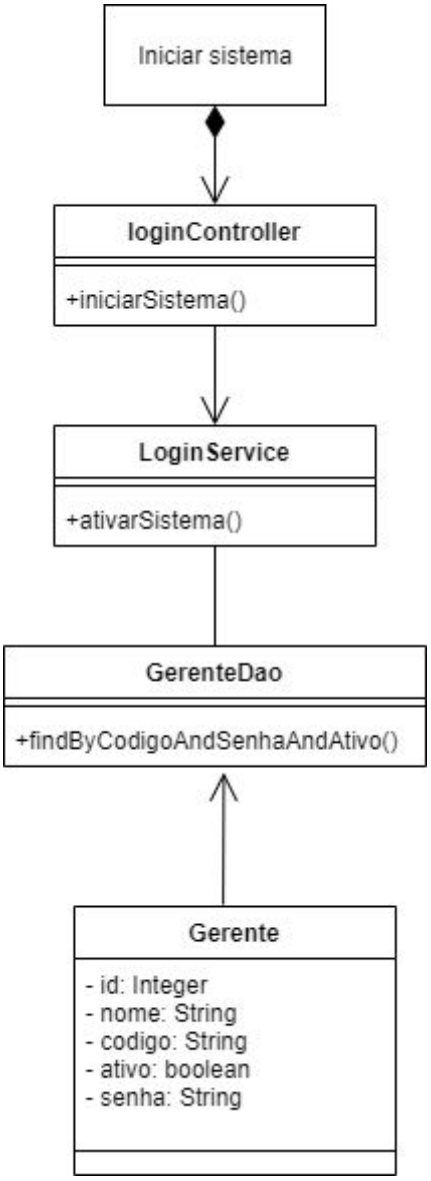


5.4 VISÃO DE CLASSES PARTICIPANTES

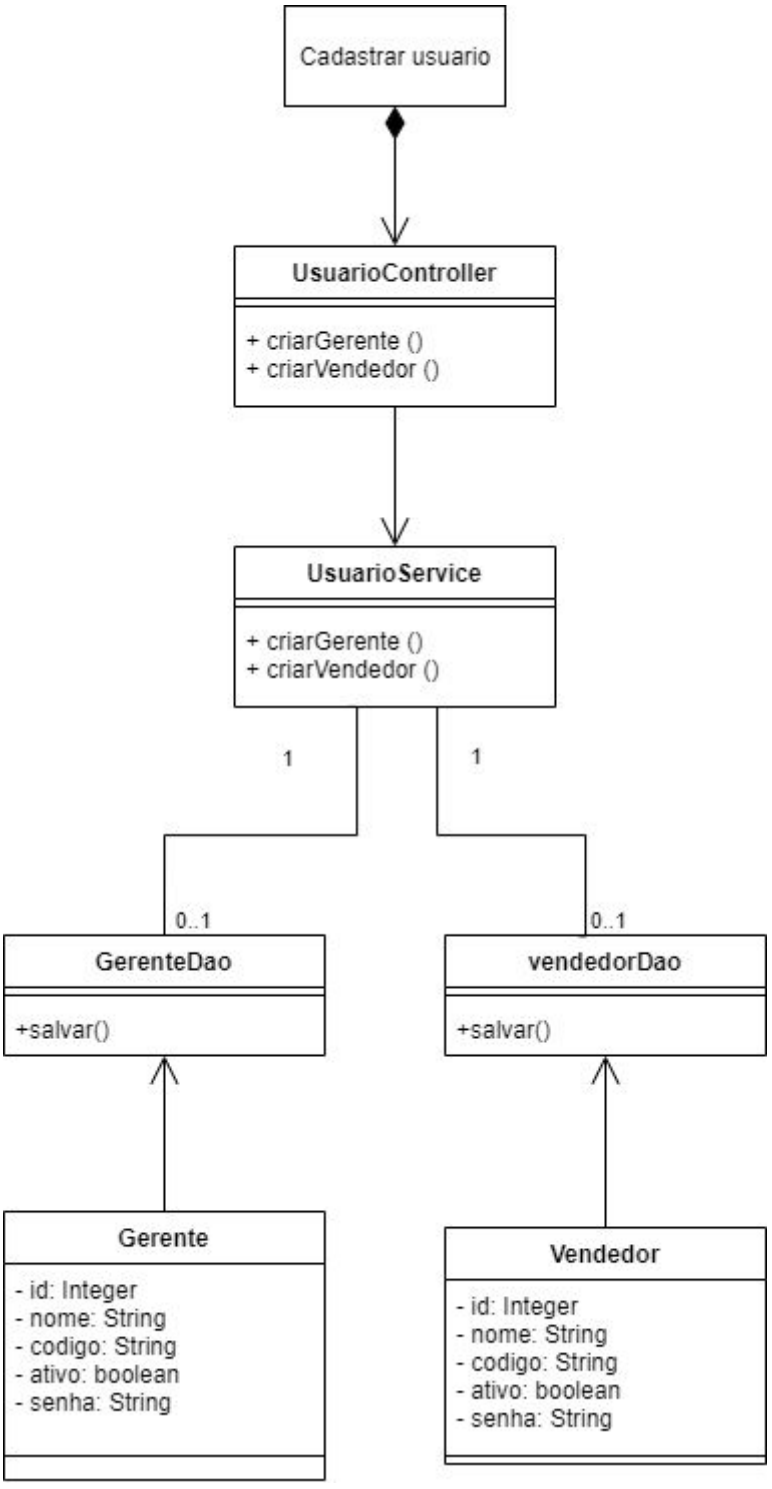
5.4.1 LOGAR NO SISTEMA



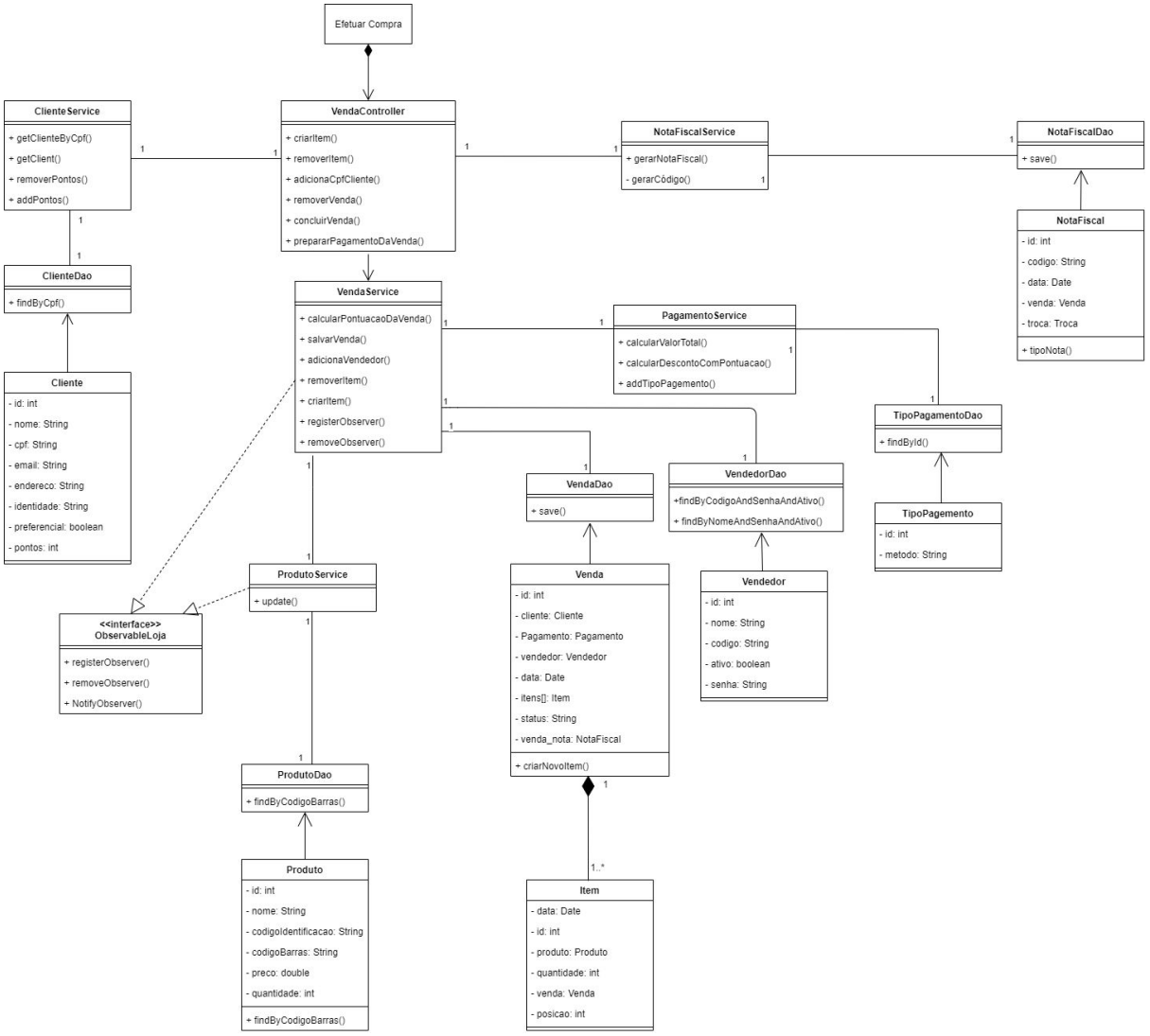
5.4.2 INICIAR SISTEMA



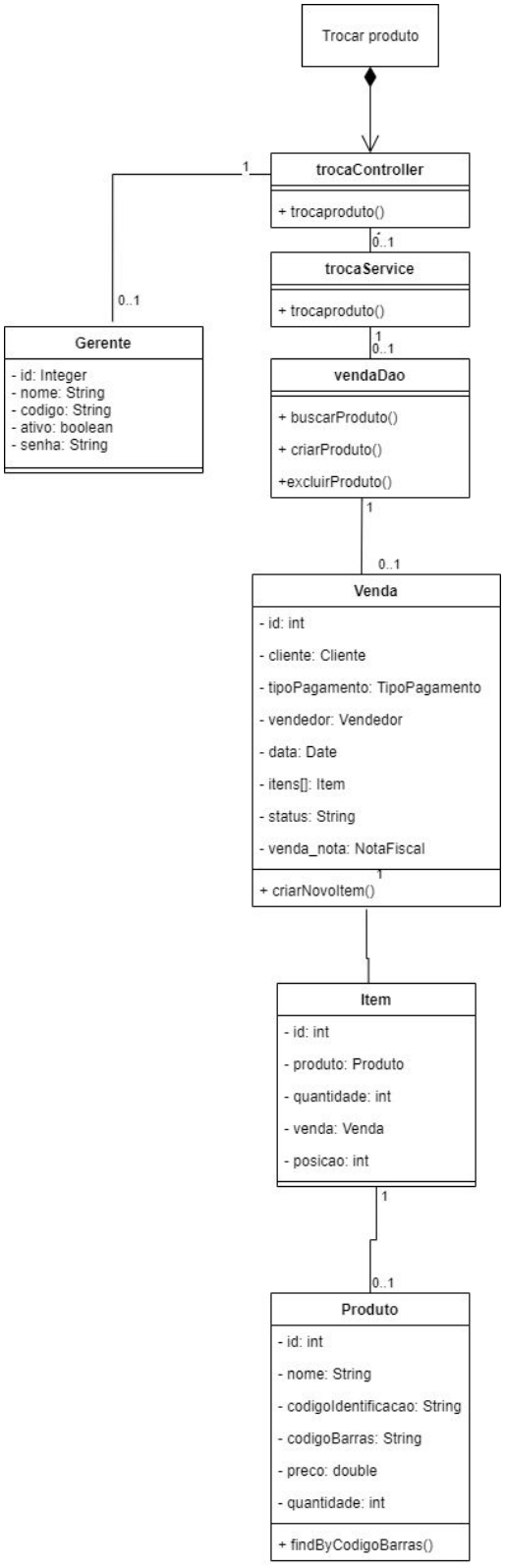
5.4.3 CADASTRAR USUÁRIO



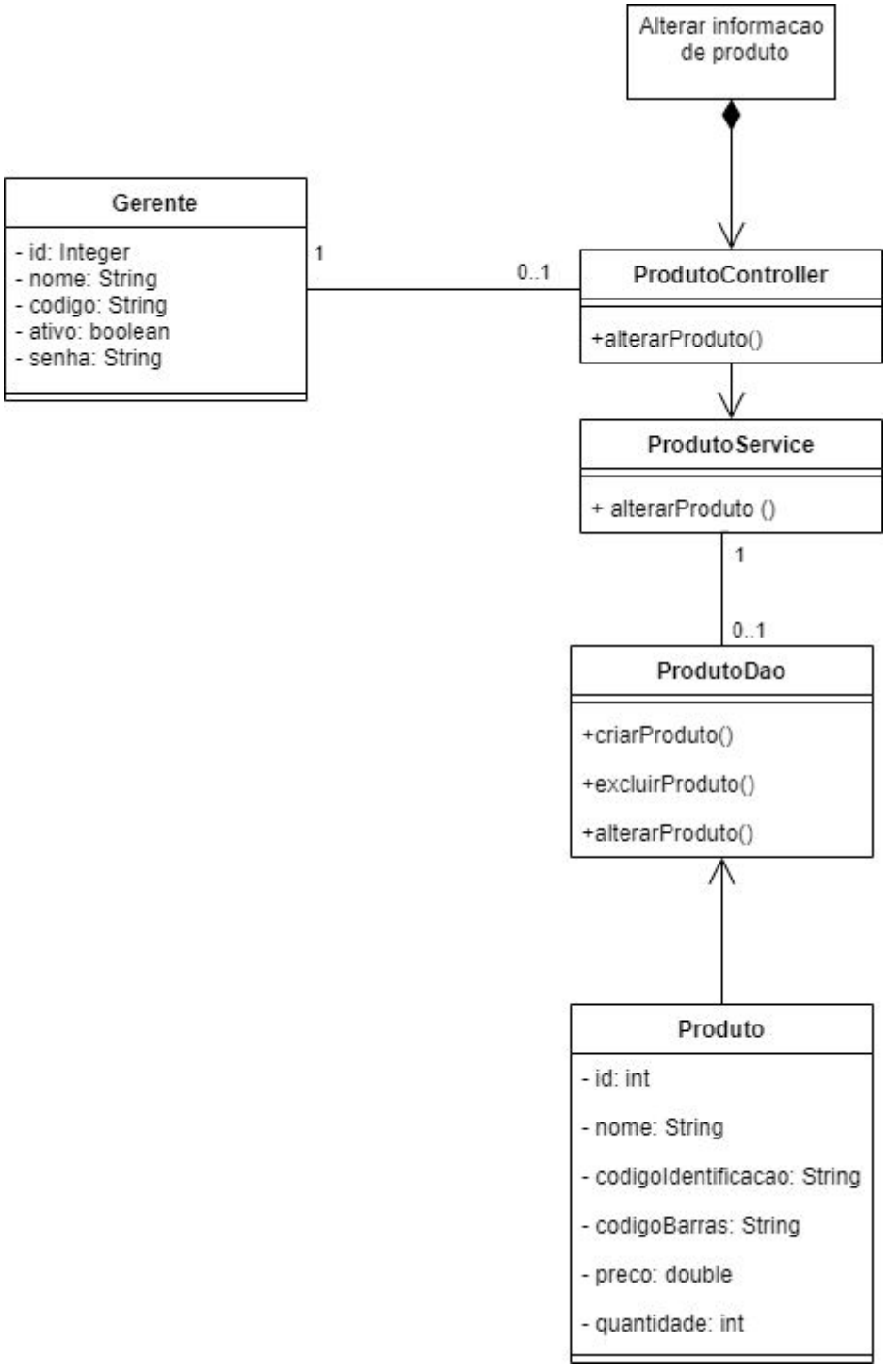
5.4.4 EFETUAR COMPRA



5.4.5 TROCAR PRODUTO

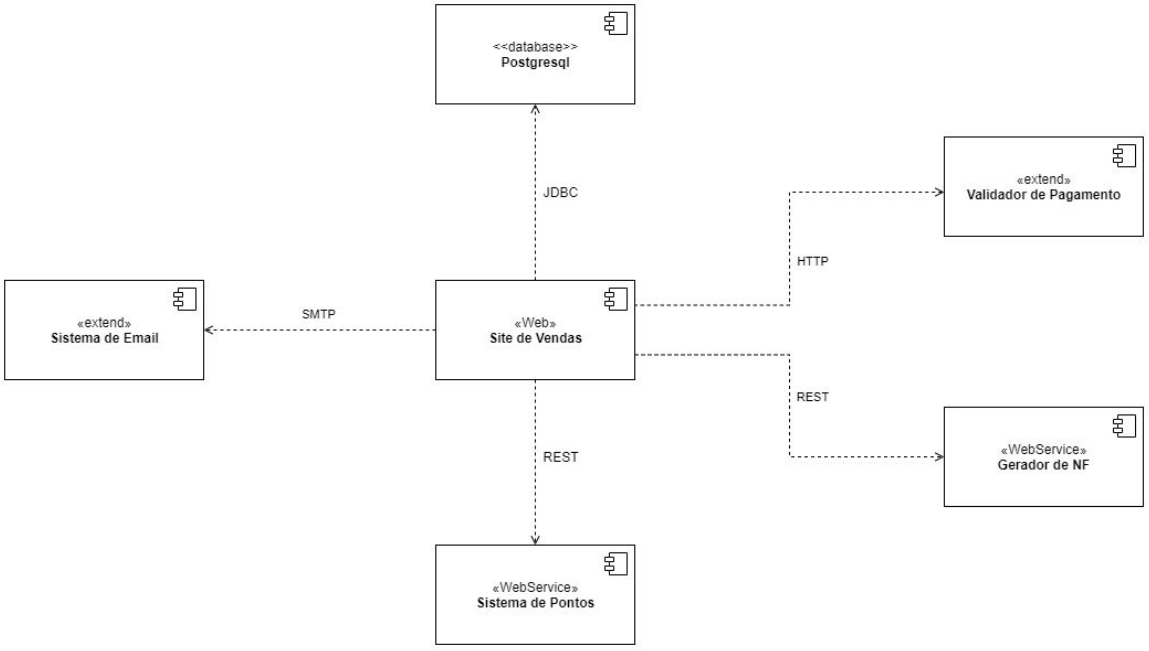


5.4.6 ALTERAR INFORMAÇÕES DE PRODUTOS

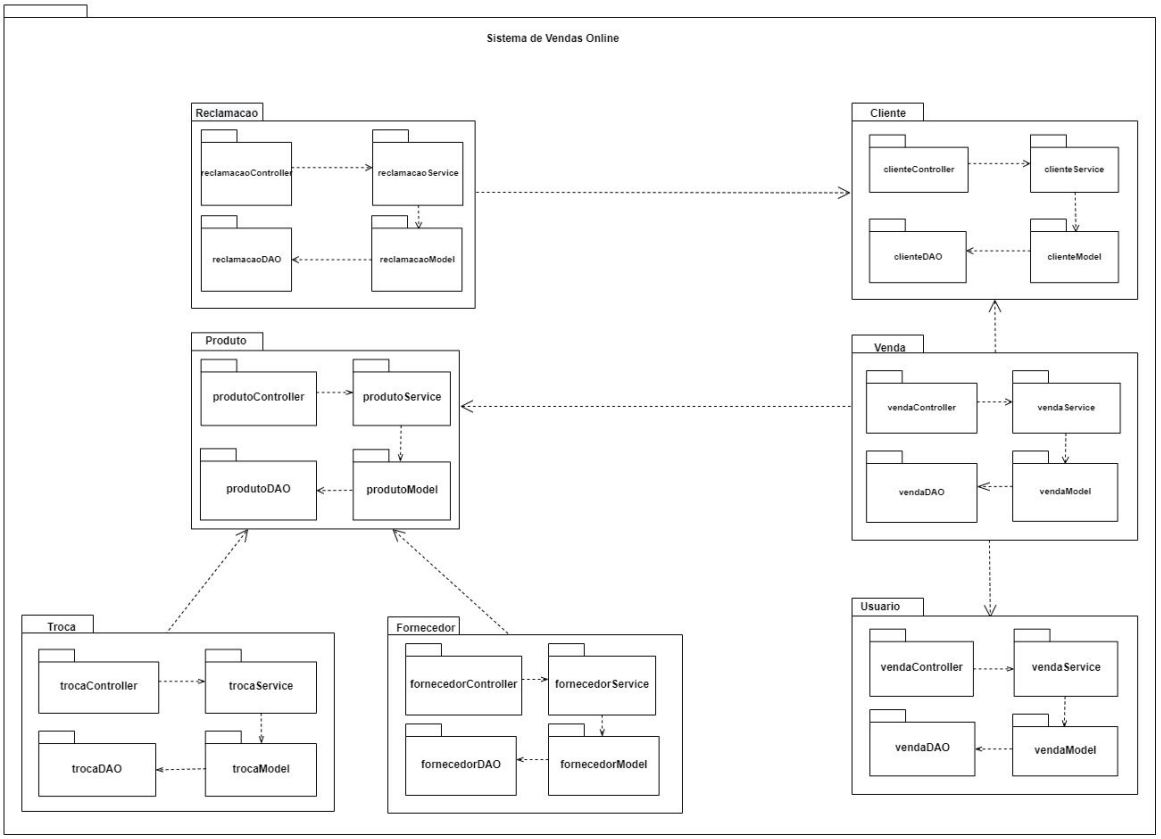


6 VISÃO DE DESENVOLVIMENTO

6.1 DIAGRAMA DE COMPONENTES

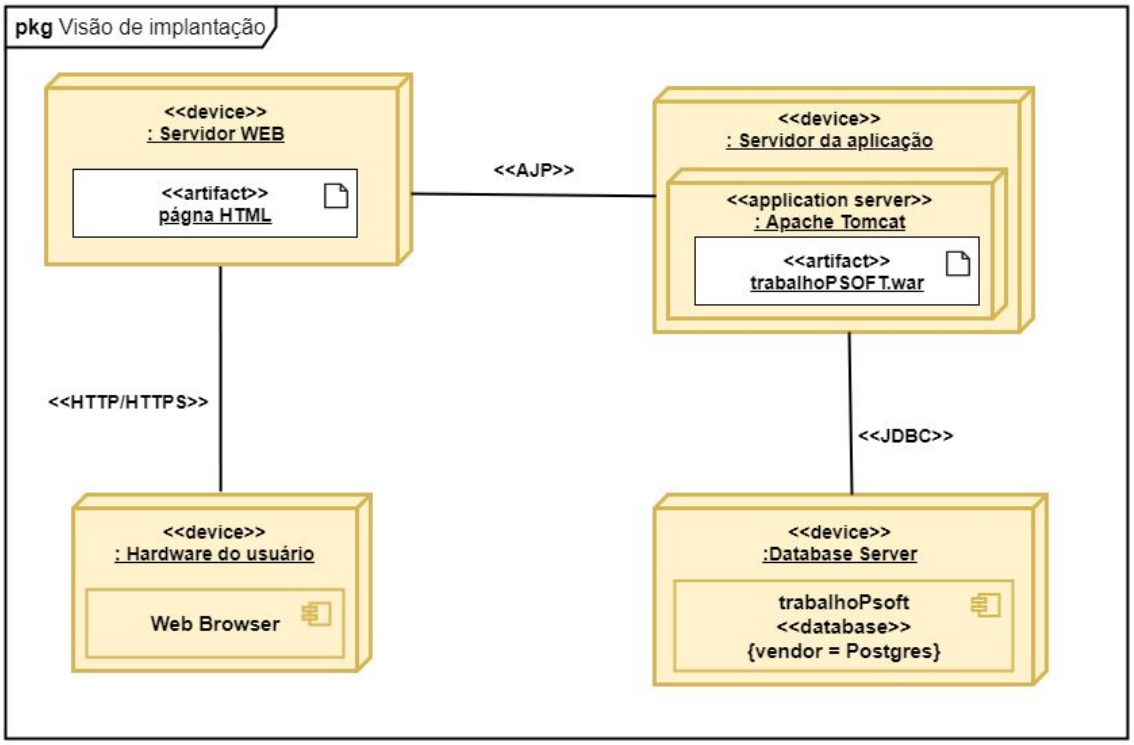


6.2 DIAGRAMA DE PACOTES



7 VISÃO FÍSICA

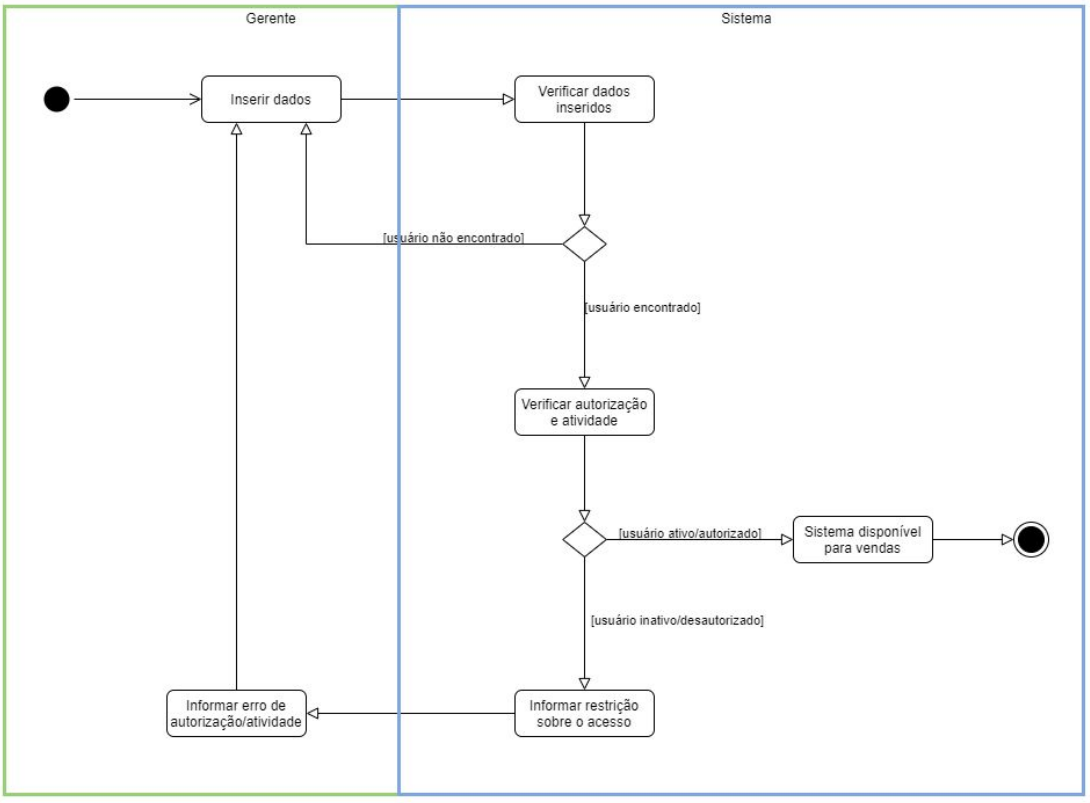
7.1 DIAGRAMA DE IMPLANTAÇÃO



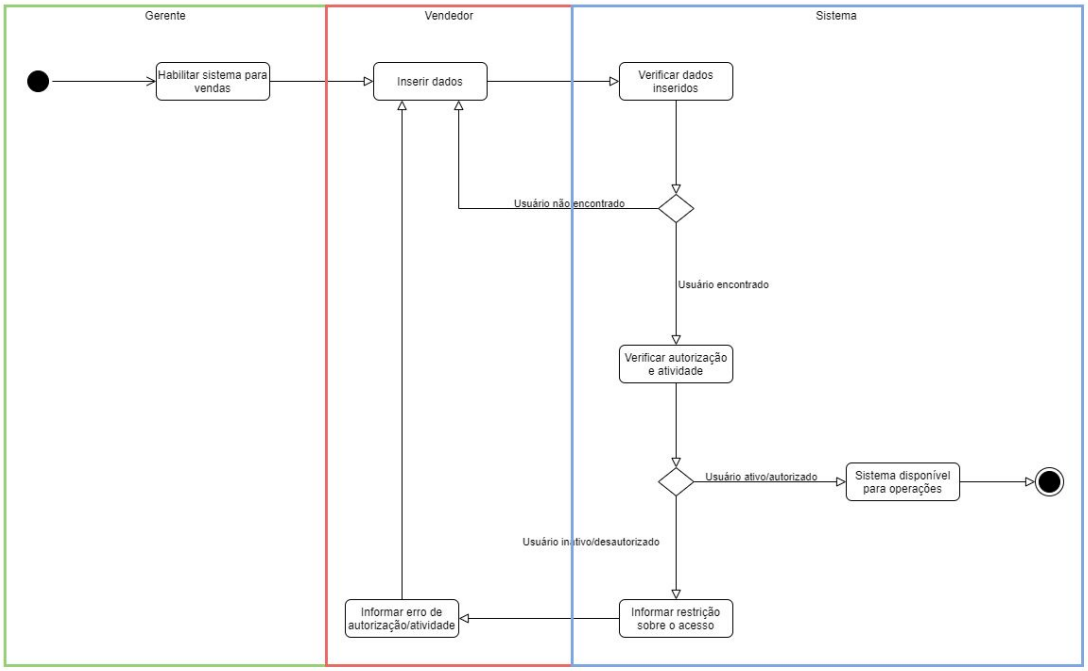
8 VISÃO DE PROCESSOS

8.1 DIAGRAMA DE ATIVIDADE

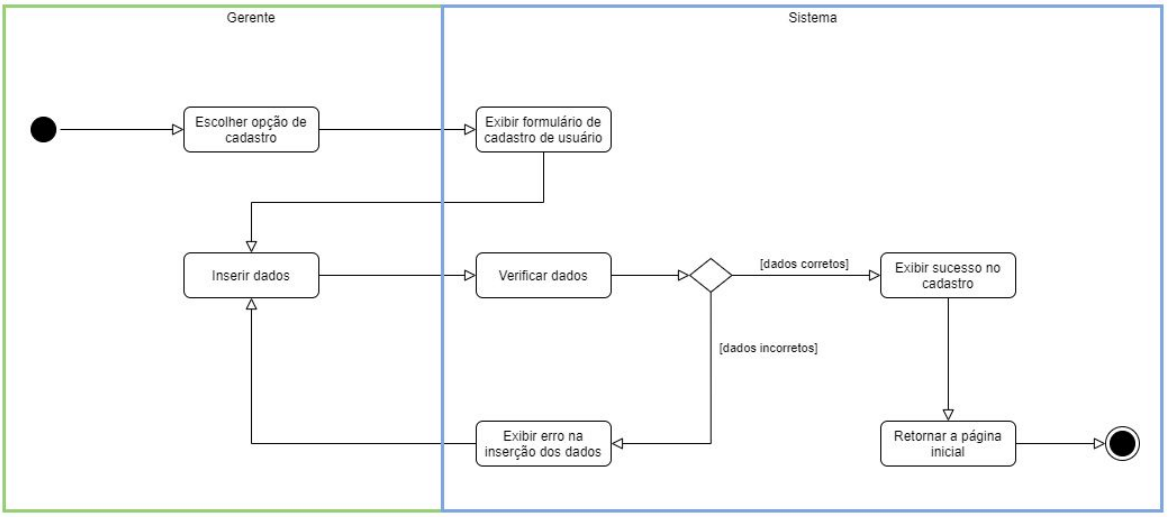
8.1.1 LOGAR NO SISTEMA



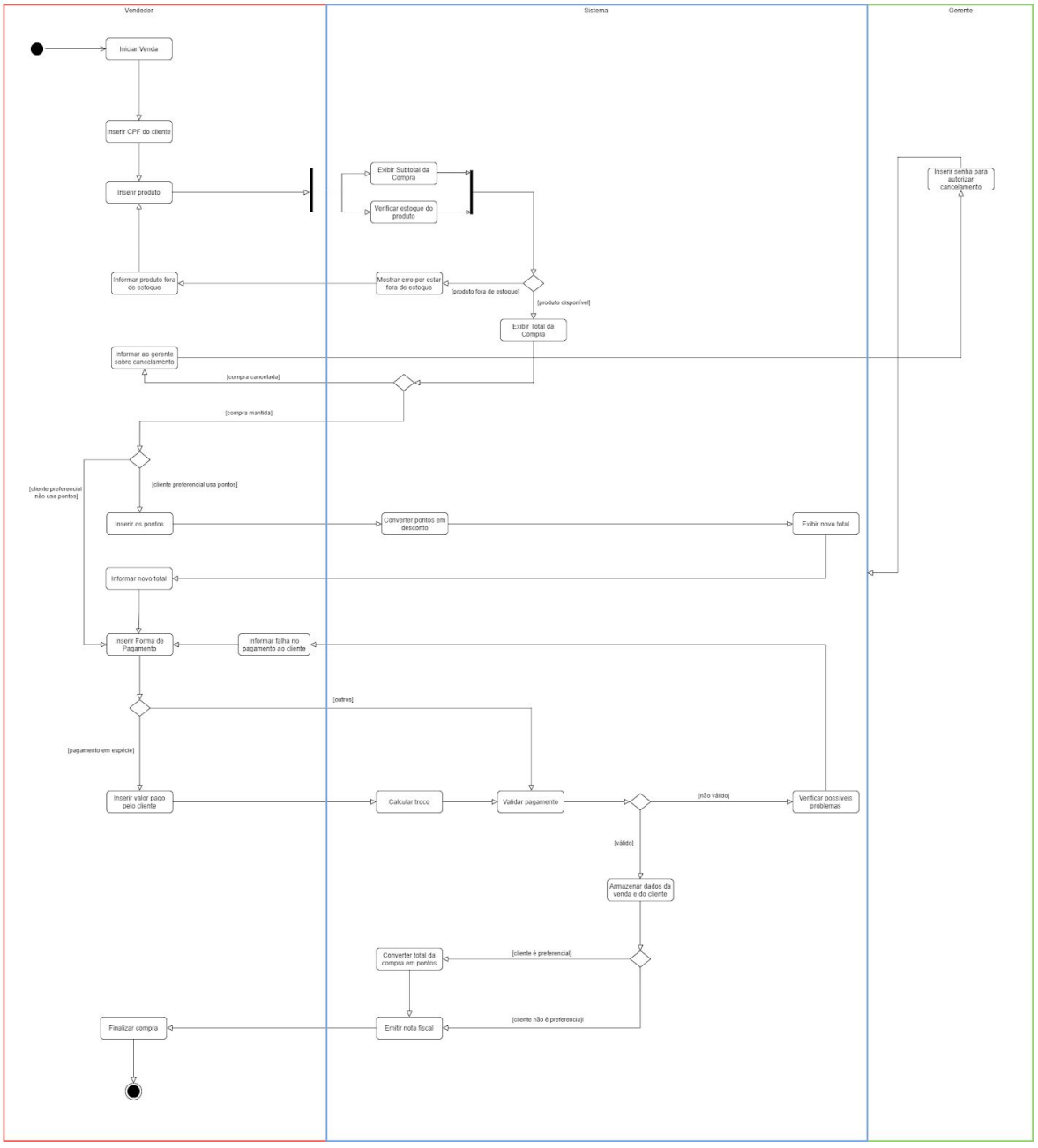
8.1.2 INICIAR SISTEMA



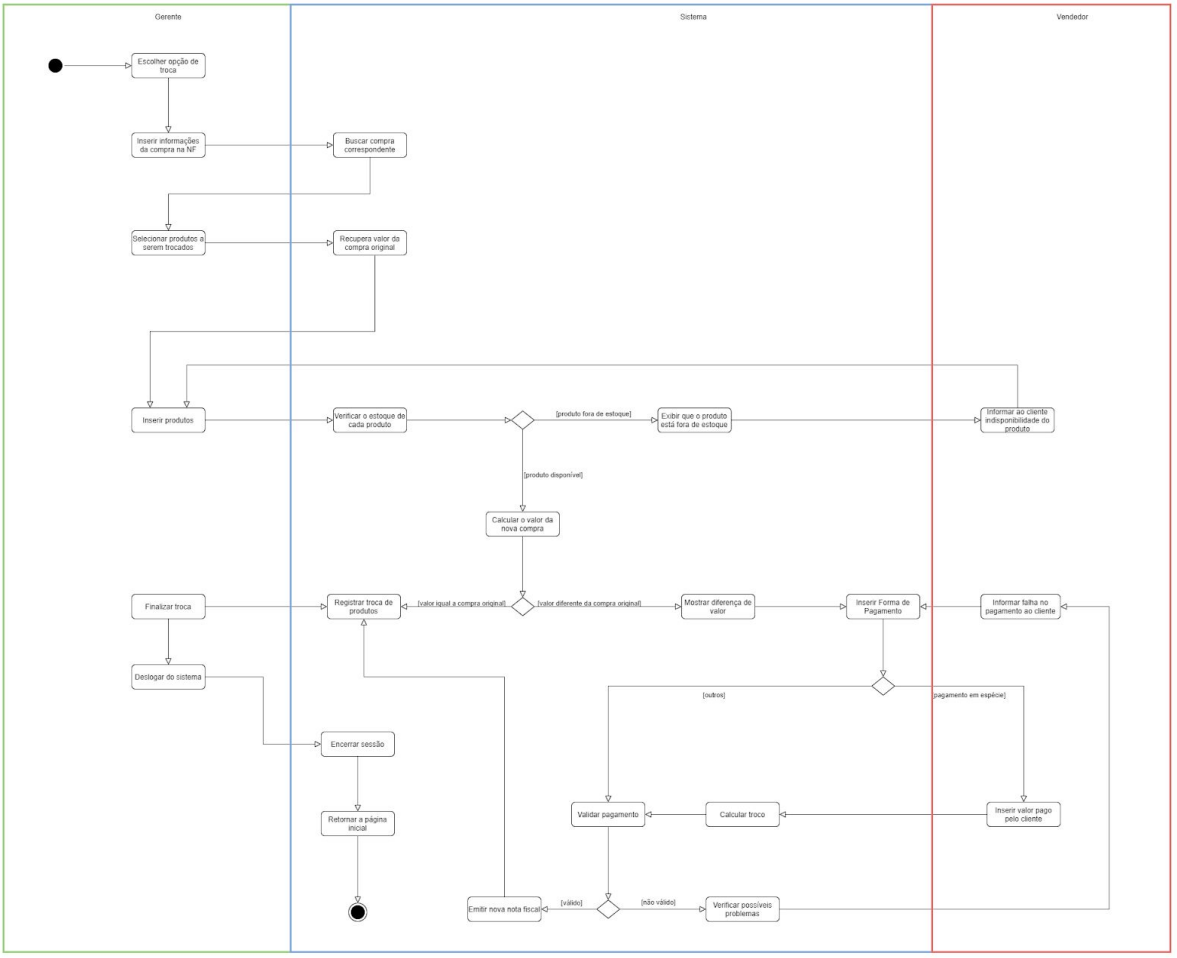
8.1.3 CADASTRAR USUÁRIO



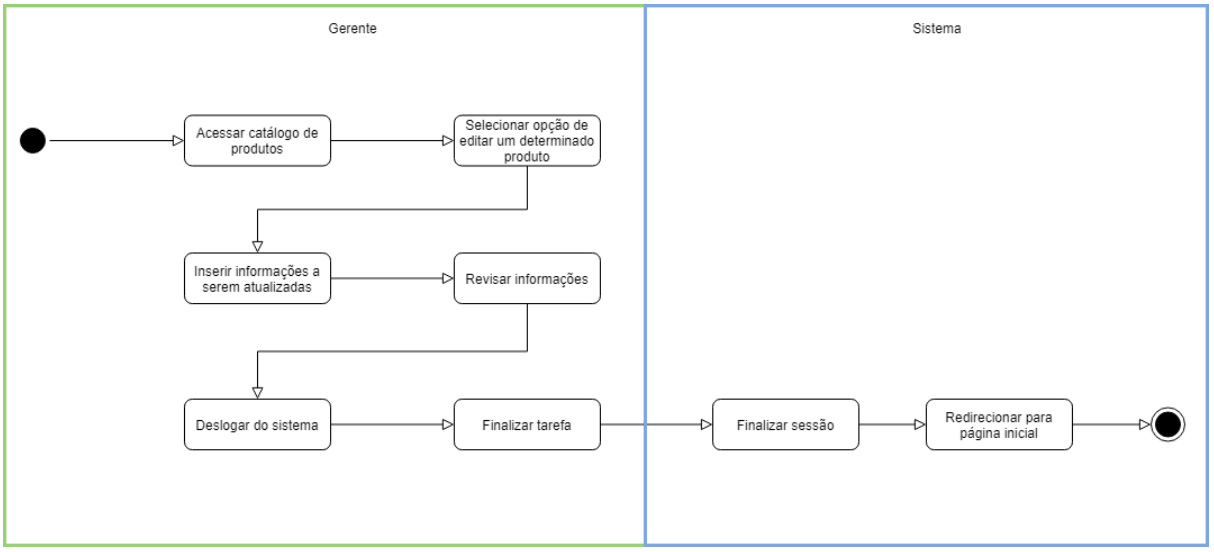
8.1.4 EFETUAR COMPRA



8.1.5 TROCAR PRODUTO

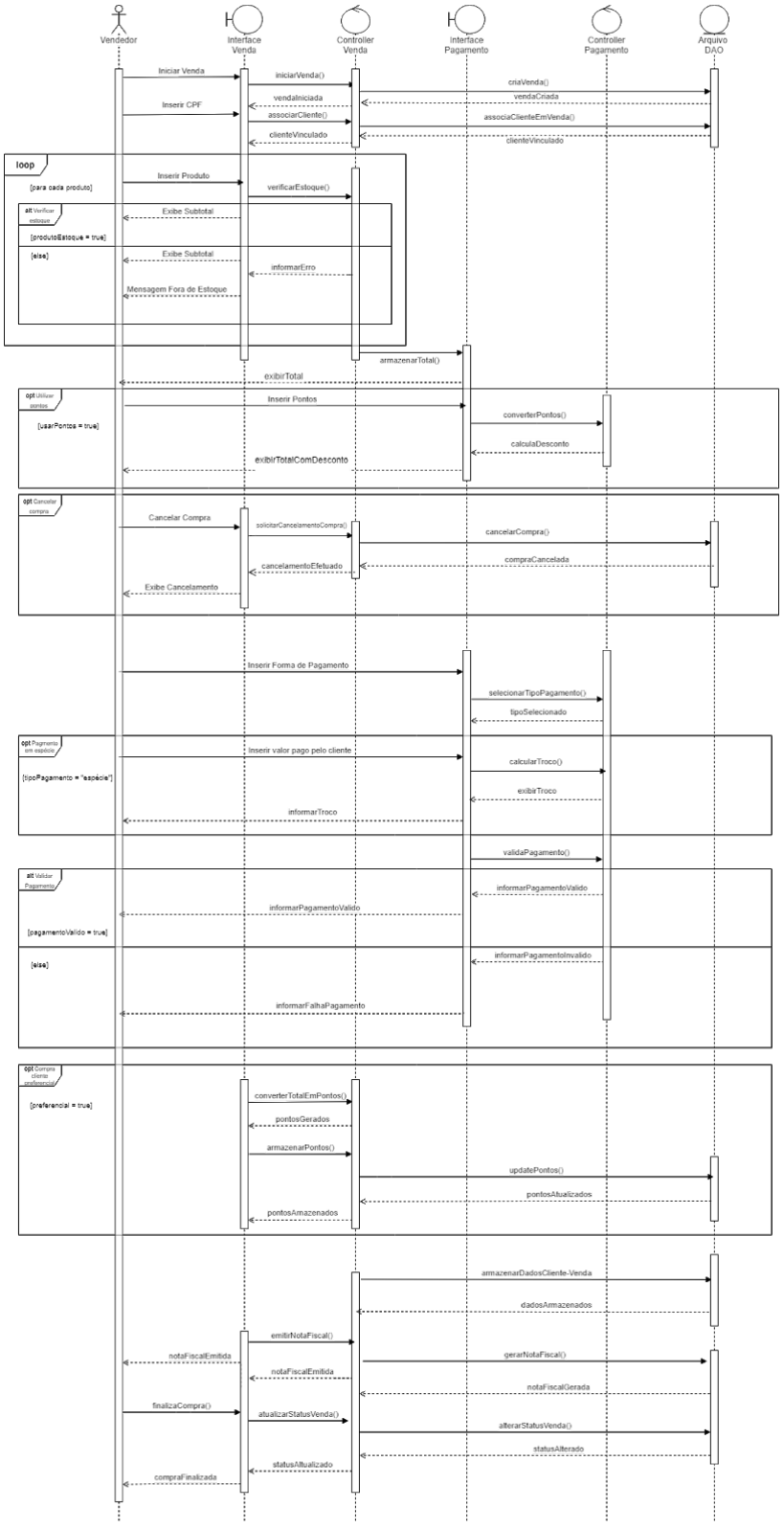


8.1.6 ALTERAR INFORMAÇÕES DE PRODUTOS

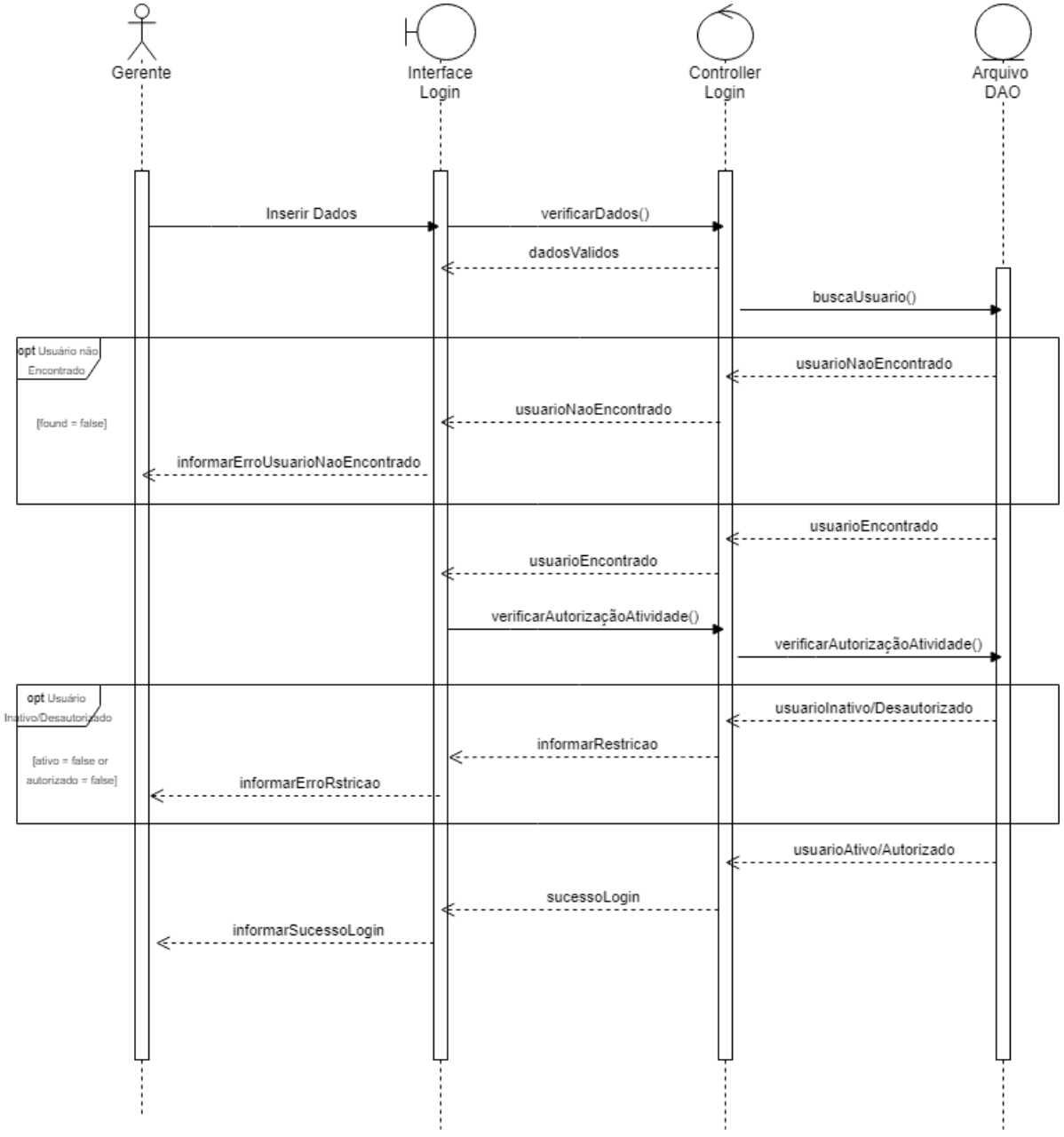


8.2 DIAGRAMA DE SEQUÊNCIA

8.2.1 EFETUAR COMPRA



8.2.2 LOGAR NO SISTEMA



9 DETALHAMENTO DOS PADRÕES

9.1 PADRÕES DE PROJETO

9.1.1 MVC

MVC é um padrão de projeto dividido em três camadas interconectadas, onde a apresentação dos dados e interação dos usuários são separados dos métodos que interagem com o banco de dados. Essa separação em camadas ajuda na redução de acoplamento e promove o aumento de coesão nas classes do projeto, facilitando a manutenção do código e a sua reutilização.

Cada uma das camadas Model, View e Controller, executa apenas o que lhe foi definido. A camada Model consiste na parte lógica da aplicação e gerencia o comportamento dos dados baseado nas regras de negócio. Se preocupa apenas com o armazenamento, manipulação e geração de dados.

A camada View é onde os dados do Model são exibidos, é a interface gráfica com o usuário, utilizada para receber a entrada de dados e apresentar visualmente o resultado. É importante que a View não acesse diretamente os dados do Model.As entradas de dados geram um evento que será tratado na camada Controller.

A camada Controller faz a mediação entre o Model e a View, manipulando os dados a serem alterados de forma apropriada conforme a solicitação do usuário pelos eventos gerados na View. O Controller interpreta esses eventos e envia ação a ser feita para o Model, e devolve o resultado para a View. O controller então é responsável por enviar comandos para atualizar o estado dos dados no Model ou alterar a apresentação desses dados na View.

Composição MVC do projeto, constitui-se da seguinte maneira: View, Controller, Service e Model (DAO). Tal organização nos auxiliou no desenvolvimento mais padronizado e determinado, além de facilitar o entendimento do código pelos integrantes do grupo.

9.1.2 OBSERVER

O Observer é um padrão de projeto que define uma dependência entre objetos de modo que quando um objeto muda o estado, todos seus dependentes são notificados e atualizados automaticamente. Este padrão deve ser usado quando uma mudança a um objeto requer mudanças a outros e você não sabe quantos outros objetos devem mudar ou quando um objeto deve ser capaz de avisar outros sem fazer suposições sobre quem são os objetos. Em outras palavras, sem criar um acoplamento forte entre os objetos.

O objeto chamado Subject mantém uma lista de seus dependentes, chamados de Observers, e os notifica automaticamente de eventuais mudanças de estado, geralmente, chamando um dos seus métodos. Assim este padrão define a criação de dois atores principais: o objeto sendo observado (o subject, ou observable) e o objeto observador (observer , ou listener), ambos com interfaces respectivas para registrar os métodos obrigatórios.

Durante o desenvolvimento, o Observer foi utilizado no mecanismo de venda, em que toda vez que esse tipo de operação é executada, o sistema irá verificar os itens presentes nela e atualizar o estoque do produto, de acordo.

9.1.3 DAO

Data Acess Object, mais conhecido como DAO, é um padrão para aplicações que utiliza, persistência de dados, onde existe a separação das regras de negócio das regras de acesso ao banco de dados por meio de uma interface. É geralmente usada em linguagens orientadas a objetos com integração da arquitetura MVC. Este padrão permite que possamos mudar a forma de persistência sem que isso influencie em nada na lógica de negócio, além de tornar nossas classes mais legíveis.

As classes DAO são responsáveis por trocar informações com o SGBD e fornecer operações CRUD e de pesquisas, elas são capazes de buscar dados no banco e transformar esses em objetos ou lista de objetos, e também receber os objetos, converter em instruções SQL e mandar para o banco de dados. Toda interação com a base de dados deve ocorrer através destas classes, abstraindo completamente o modo de busca e gravação de dados, dando transparência para a aplicação. Essas características facilitam mudanças futuras na aplicação ou uma possível migração de banco de dados.

O DAO tem como papel no sistema de receber as funções acionadas e gravar os resultados das mesmas no banco de dados determinado, que no nosso caso o escolhido foi o PostgreSQL 13.1.

9.1.4 SERVICE LAYER

O SERVICE LAYER é um padrão que tem como objetivo realizar uma organização dos serviços dentro de um conjunto de vários deles, os assentando em camadas lógicas. Tal mecanismo auxilia na redução da sobrecarga conceitual no que diz respeito ao gerenciamento dos serviços, já que os serviços presentes na mesma camada tendem a tratar de um conjunto menor de atividades.

A possibilidade da união dos serviços em camadas funcionais reduz exponencialmente o impacto das mudanças sobre a aplicação, em que geralmente as alterações implicam somente nas camadas em que foram feitas, sem maiores consequências para as demais camadas. Esses benefícios proporcionam ao código duas características importantes e valiosas, que são a manutenção e a reutilização dos seus serviços.

Em relação a esse padrão de projeto, ele foi responsável por isolar e abstrair as regras de negócio da parte denominada como Controller, o que organiza o projeto e impede justamente a sobrecarga conceitual relatada anteriormente.

9.1.5 Grasp - EXPERT

Expert ou Especialista na Informação é um padrão de projeto pertencente ao conjunto de princípios denominado GRASP. É um entendimento mais generalizado sobre atribuir a responsabilidade de uma tarefa ou de uma informação ao elemento de fato conhecimento como "especialista da informação", a classe que possui os requisitos necessários para cumprir tal responsabilidade.

Nesse caso o padrão EXPERT foi usado em algumas das classes do sistema, que possuíram o papel de especialista, como a Venda.java, Produto.java, Gerente.java, Vendedor.java, cada qual responsável por suas informações.

9.1.6 Grasp - CREATOR

Creator também é um padrão de projeto pertencente ao conjunto GRASP, que tem como objetivo estipular a classe que será encarregada por criar uma instância nova de uma outra. A criação de objetos por ser uma das atividades mais comuns no sistema orientado a objetos, é extremamente útil ter um princípio definido para traçar os papéis de cada classe nesse sentido.

Em relação ao CREATOR, havia a classe Venda.java, que possuía os itens da da venda e então criava os itens a partir dos dados de cada produto correspondente, instanciando essas novas classes de Item.java.

9.1.7 INTERCEPTOR

Interceptor é um padrão de design de software usado quando sistemas de software ou estruturas desejam oferecer uma maneira de alterar ou aumentar seu ciclo de processamento usual. Por exemplo, uma sequência de processamento típica (simplificada) para um servidor da web é receber um URI do navegador, mapeá-lo para um arquivo em disco, abrir o arquivo e enviar seu conteúdo ao navegador. Qualquer uma dessas etapas pode ser substituída ou alterada, por exemplo, substituindo a maneira como os URIs são mapeados para nomes de arquivos ou inserindo uma nova etapa que processa o conteúdo dos arquivos.

O padrão de projeto em questão teve como papel, como o próprio nome sugere, interceptar as urls a serem enviadas pelo controller, realizando dois tipos de verificações. A primeira é se o sistema está de fato ativo para vendas, caso negativo há um redirecionamento para uma página de ativar o sistema, caso positivo, o fluxo é continuado normalmente. Já a outra verificação é se possui um usuário logado para que as tarefas possam ser realizadas, caso negativo também há um redirecionamento para a página login, caso positivo o fluxo prossegue como o esperado.

10 CONCLUSÃO

Portanto, após a finalização das etapas de desenvolvimento o que se pode absorver foi como aplicar padrões de projeto de forma consistente e útil no código, colocando em prática os conhecimentos retidos em relação ao campo de engenharia de software. Tais aprendizados foram impulsionados pela variedade de padrões selecionados e implementados no sistema de fato, como por exemplo pode-se notar os benefícios aparentes ao se utilizar o padrão arquitetural Model, View e Controller.

Quanto ao desfecho da documentação, foram incorporados novos diagramas ao repertório dos integrantes do grupo, possibilitando assim tanto uma ampliação da visão de apresentação do sistema quanto o aprofundamento nas diferentes formas de entendimento da aplicação. De acordo com essa perspectiva, como exemplo, foi visto que o diagrama de representação arquitetural torna visível de forma geral a arquitetura planejada para o desenvolvimento, ajudando tanto os integrantes responsáveis pelo desenvolvimento quanto a outros stakeholders, como investidores, o que torna o projeto mais sólido e coeso.