**Café Mania**

**Documento de Arquitetura de Software**

**Versão <1.2.0>**

*[Observação: O template a seguir é fornecido para uso com o Rational Unified Process (RUP). O texto em azul exibido entre colchetes e em itálico (style=InfoBlue) foi incluído para orientar o autor e deve ser excluído antes da publicação do documento. Um parágrafo digitado após esse estilo será automaticamente definido como normal (style=Body Text).]*

*[Para personalizar campos automáticos no Microsoft Word (que exibem um fundo cinza quando selecionados), escolha File>Properties e substitua os campos Title, Subject e Company pelas informações apropriadas para este documento. Depois de fechar a caixa de diálogo, para atualizar os campos automáticos no documento inteiro, selecione Edit>Select All (ou Ctrl-A) e pressione F9 ou simplesmente clique no campo e pressione F9. Isso deve ser feito separadamente para Cabeçalhos e Rodapés. Alt-F9 alterna entre a exibição de nomes de campos e do conteúdo dos campos. Consulte a ajuda do Word para obter mais informações sobre como trabalhar com campos.]*

**Histórico da Revisão**

| **Data** | **Versão** | **Descrição** | **Autor** |
| --- | --- | --- | --- |
| 17/05/2022 | 1.0.0 | Primeira versão do documento de arquitetura de software seus respectivos requisitos, estilos arquiteturais e modelos VCP. | Grupo 1 |
| 16/06/2022 | 1.1.0 | Refinamento dos modelos VCP e casos de uso, adição do diagrama de classes e diagramas de sequência e justificativa dos modelos arquiteturais escolhidos. | Grupo 1 |
| 21/06/2022 | 1.2.0 | Adição dos padrões de projeto GOF e os seus respectivos modelos de instanciação. | Grupo 1 |
| 30/06 | 1.2.1 | Refatoração de alguns padrões de projeto, solucionando problemas na aplicação. | Grupo 1 |

**Índice Analítico**

[**Introdução**](#_gqfnzr3ygbkz) **5**

[**Finalidade**](#_ylql1lqhll6w) **5**

[**Metas e Restrições da Arquitetura**](#_cr3og8jyw8yj) **5**

[**Suposições e Dependências**](#_w98nzaueaa29) **6**

[**Requisitos Arquiteturalmente Significantes**](#_f3dp4lmvcher) **6**

[**Decisões, Restrições e justificativas**](#_lwk9rsyje1kl) **6**

[**Mecanismos Arquiteturais**](#_gmkcpxnuaa8j) **6**

[**Camadas da Arquitetura**](#_eq2ehzawsw1) **8**

[**Visões da Arquitetura**](#_a6emhxe17fqs) **8**

[**Visão dos casos de uso**](#_ei95ejwb5qx7) **9**

[**Visões das classes participantes**](#_uplgsxsrhbim) **9**

[**Caso de Uso Efetuar Venda**](#_n54s5d6tz58v) **10**

[**Caso de Uso Emitir Relatório**](#_s6oibklog1g3) **11**

[**Caso de Uso Manter Cliente**](#_z7no27g3k6oc) **12**

[**Caso de Uso Manter Produto**](#_kryhtjc3d0ib) **13**

[**Caso de Uso Realizar Login**](#_74xz59h0dt92) **14**

[**Visão Lógica**](#_mu9o6ensigx1) **14**

[**Diagrama de Pacotes da Arquitetura**](#_cica0yevbh5w) **15**

[**Diagrama de Classes da Arquitetura**](#_oizwooocvujn) **16**

[**Diagramas de Sequência**](#_sw26cgd62sp9) **17**

[**Caso de Uso Efetuar Venda**](#_qt8uk94njcg4) **17**

[**Caso de Uso Emitir Relatório**](#_b1bejmwmahdi) **18**

[**Caso de Uso Manter Cliente**](#_eies26l9rs9z) **19**

[**Caso de Uso Manter Produto**](#_rp9n58z3lp5r) **20**

[**Caso de Uso Realizar Login**](#_6c9x55cj6xnr) **21**

[**Qualidade**](#_r1t4qo2y7aul) **21**

**Documento de Arquitetura de Software**

# Introdução

*[A introdução do* ***Documento de Arquitetura de Software*** *fornece uma visão geral do documento inteiro. Ela inclui a finalidade, o escopo, as definições, os acrônimos, as abreviações, as referências e a visão geral do* ***Documento de Arquitetura de Software****.]*

Este trabalho tem como objetivo modelar e projetar um software de gerenciamento de uma cafeteria, a “Café com Serviços”. Ao longo deste documento, pode ser encontrado as descrições das Visões de Arquitetura, classes e as decisões que nortearam este projeto.

# Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

*[Esta seção define o papel ou finalidade do* ***Documento de Arquitetura de Software****, na documentação do projeto como um todo, e descreve rapidamente a estrutura do documento. O público-alvo específico do documento é identificado, com uma indicação de como ele espera usar o documento.]*

# Metas e Restrições da Arquitetura

*[Descreva a filosofia da arquitetura. Identifique as questões que dirigem a arquitetura, como “O sistema será dirigido por funcionalidades complexas, integração com sistemas legados ou questões de desempenho? O sistema precisa ser robusto durante um longo tempo de manutenção ?*

*[Formule um conjunto de metas que a arquitetura tem de cumprir em sua estrutura e comportamento. Identifique questões críticas que a arquitetura deve resolver, tais como “há dependências de hardware que precisam ser isoladas o resto do sistema?” e “O sistema precisa funcionar de forma eficiente em condições não-usuais?”]*

A arquitetura de projeto utilizada neste trabalho é a Arquitetura MVC. Tal padrão de projeto organiza as classes do programa em três eixos: As de Visão, que são classes responsáveis pela apresentação da interface gráfica do sistema; as de Controle, que definem o comportamento da aplicação e que interpretam as ações do usuário e as mapeiam para chamadas das classes de Modelo, essas últimas que são responsáveis por gerenciar e controlar a forma como os dados se comportam por meio das funções, lógicas e regras de negócios estabelecidas.

A comunicação entre interfaces e regras de negócios é definida através de um controlador, que separa as camadas. Quando um evento é executado na interface gráfica, como um clique em um botão, a interface se comunicará com o controlador, que por sua vez se comunica com as regras de negócios.

Dado que a aplicação precisa se comunicar com um banco de dados MySQL para facilitar o acesso e gerenciamento dos dados, foi utilizado o estilo arquitetural Shared Data, com o objetivo de manter todas as informações residentes em um único ambiente.

Outro ponto, é que como o objetivo do sistema é realizar a maior quantidade de execuções e com a menor quantidade de erros possíveis, também foi escolhido o estilo arquitetural de microsserviços. Por isso, a parte do sistema responsável pelo backend funcionará de forma independente das demais, assim como as partes responsáveis pelo frontend e pelas manipulações do banco de dados.

E por último, a arquitetura cliente-servidor. Esta arquitetura será empregada uma vez que a sessão do sistema responsável pelo gerenciamento e armazenamento dos dados funcionará como um servidor, estando permanentemente online e que responderá às requisições das classes do frontend do sistema, essas últimas que terão o comportamento de cliente.

# Suposições e Dependências

*[Liste as suposições e dependências que dirigem as decisões arquiteturais. Isto pode incluir áreas sensíveis ou críticas, dependências e interfaces com sistemas legado, a habilidade e experiência da equipe, a disponibilidade de recursos importantes, e assim por diante ]*

# Requisitos Arquiteturalmente Significantes

*[Insira uma referência ou link para os requisitos que exploram aspectos relevantes da arquitetura.]*

# Decisões, Restrições e justificativas

*[Liste as decisões tomadas em relação às abordagens arquiteturais. Estas decisões servirão como guias para definir as partes arquiteturalmente significantes do sistema. Justifique cada decisão ou restrição para que os desenvolvedores compreendam a importância de construir o sistema dentro desse contexto. Isto pode incluir uma lista de “Faça” ou “Não Faça” para guiar os desenvolvedores no desenvolvimento do sistema]*

A abordagem presente neste trabalho tem como objetivo dividir as classes do sistema em três categorias principais, definidas na arquitetura MVC: Model, View e Controller, sendo que, as classes relacionadas a interação com o usuário estão são as Views, as relacionadas à manipulação com o banco de dados, Controllers, e por último, as Models, que são responsáveis por conectar as duas primeiras e implementar as regras de negócio. A utilização do padrão MVC traz como benefício o isolamento das regras de negócios da lógica de apresentação, que é a interface com o usuário. Isto possibilita a existência de várias interfaces com o usuário que podem ser modificadas sem a necessidade de alterar as regras de negócios, proporcionando muito mais flexibilidade e oportunidades de reuso das classes.

Dentre as razões para desenvolver a aplicação utilizando como base um padrão de projeto, é que isso pode trazer alguns benefícios, como aumento de produtividade, uniformidade na estrutura do software, redução de complexidade no código, facilita a documentação, reduz o tempo de desenvolvimento, dentre outros.

# Mecanismos Arquiteturais

*[Liste os mecanismos arquiteturais, como mecanismos de persistência, comunicação e tratamento de erros, por exemplo, e descreva o corrente estado de cada um. Inicialmente, cada mecanismo pode ser somente um nome e uma breve descrição. Eles evoluirão até que o mecanismo se torne um padrão ou uma colaboração de elementos de projeto que possam ser aplicados diretamente em algum aspecto do projeto.]*

**Mecanismo Arquitetural 1**

*[Descreva a finalidade, os atributos e funções do mecanismo arquitetural.]*

**Mecanismo Arquitetural 2**

*[Descreva a finalidade, os atributos e funções do mecanismo arquitetural.]*

# Camadas da Arquitetura

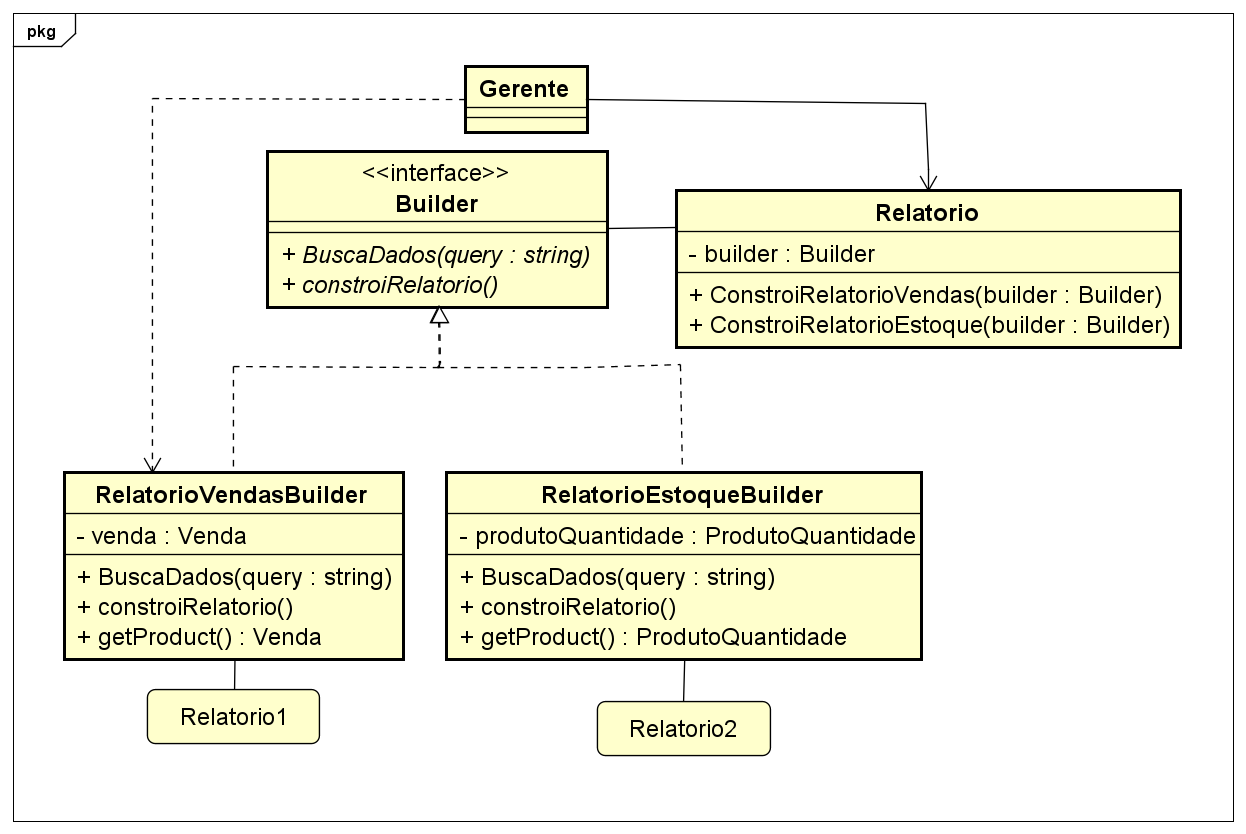
*[Descreva os padrões de arquitetura utilizados e como a arquitetura se manterá consistente e uniforme. Isto pode ser uma simples referência para um conhecido padrão arquitetural, como o padrão de divisão em camadas e uma descrição de como os componentes do sistema podem ser colocados juntos.]*

**Singleton**

A fim de controlar o acesso a recursos que são compartilhados, o padrão singleton foi adotado para a classe Usuários, para garantir que somente um usuário possa interagir com o sistema por vez, vedando que múltiplos usuários consigam acessar a base de dados simultaneamente e evitar possíveis inconsistências relacionadas ao estoque de produtos.

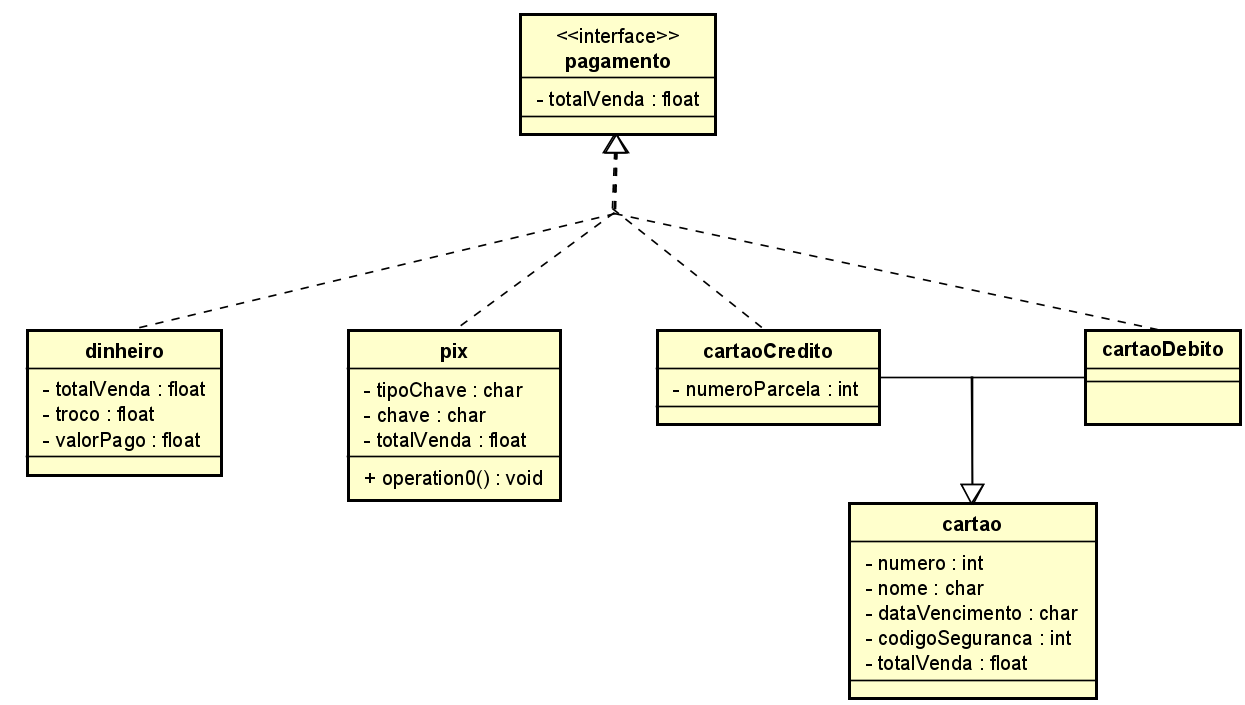
**Builder**

Uma das atribuições do sistema é a geração de diversos relatórios, e como a estrutura desses vários objetos é semelhante entre si, e com mudanças pontuais em pontos específicos do relatório. Assim, a utilização do padrão Builder se faz pertinente aqui, já que com ele podemos criar um mesmo objeto, nesse caso o relatório, com diversas opções de configurações, como a venda, data, produtos, quantidades, etc.



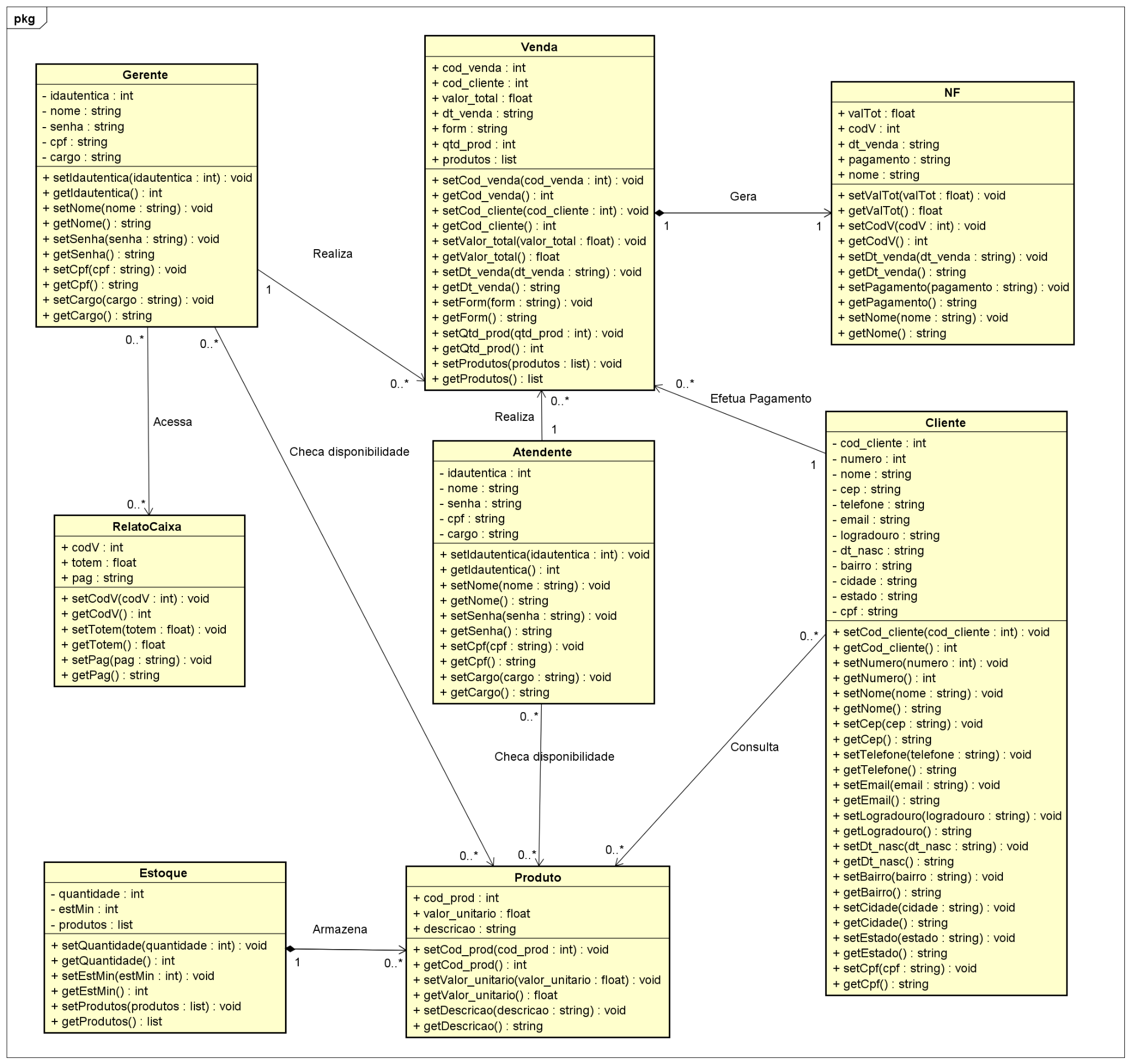
**Strategy**

O sistema, por lidar com meios de pagamentos, proporciona um ambiente que torna-se mais organizado e eficiente com a implementação do padrão Strategy. Como diferentes formas de pagamento são possíveis dentro do sistema, a interface Pagamento faz a assinatura o método pagar que irá ser implementado pelas classes PagtoDebito, PagtoCredito, PagtoValeRefeicao, PagtoPix e PagtoDinheiro com o intuito de atender as particularidades de cada meio de pagamento. Dessa forma, as responsabilidades e peculiaridades de cada forma de pagamento diferente são implementadas pelas classes que representam cada um desses mesmos pagamentos.



**DTO**

Para que houvesse uma diminuição de métodos chamados e para haver um objeto que, unicamente, aglutina e manipula os dados que vão ser carregados no banco de dados. Esse padrão vai auxiliar na transferência de dados entre as diferentes camadas da aplicação e também ajudará a reduzir a quantidade de acessos entre cliente e servidor.



**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.

**DAO**

O padrão DAO é utilizado para separar os códigos de manipulação de dados do restante da aplicação, assim facilitando na manutenção.

**BRIDGE**

O padrão Bridge é utilizado para desacoplar os códigos, assim podemos deixar as classes mais enxutas, assim, auxiliando a não haver repetição de código e também na manutenção do mesmo.

**FABRIC**

O padrão Fabric é usado para separar uma função com um único objetivo, fabricar algum objeto. No nosso caso usamos para fabricar um objeto de conexão com o banco de dados.

# Visões da Arquitetura

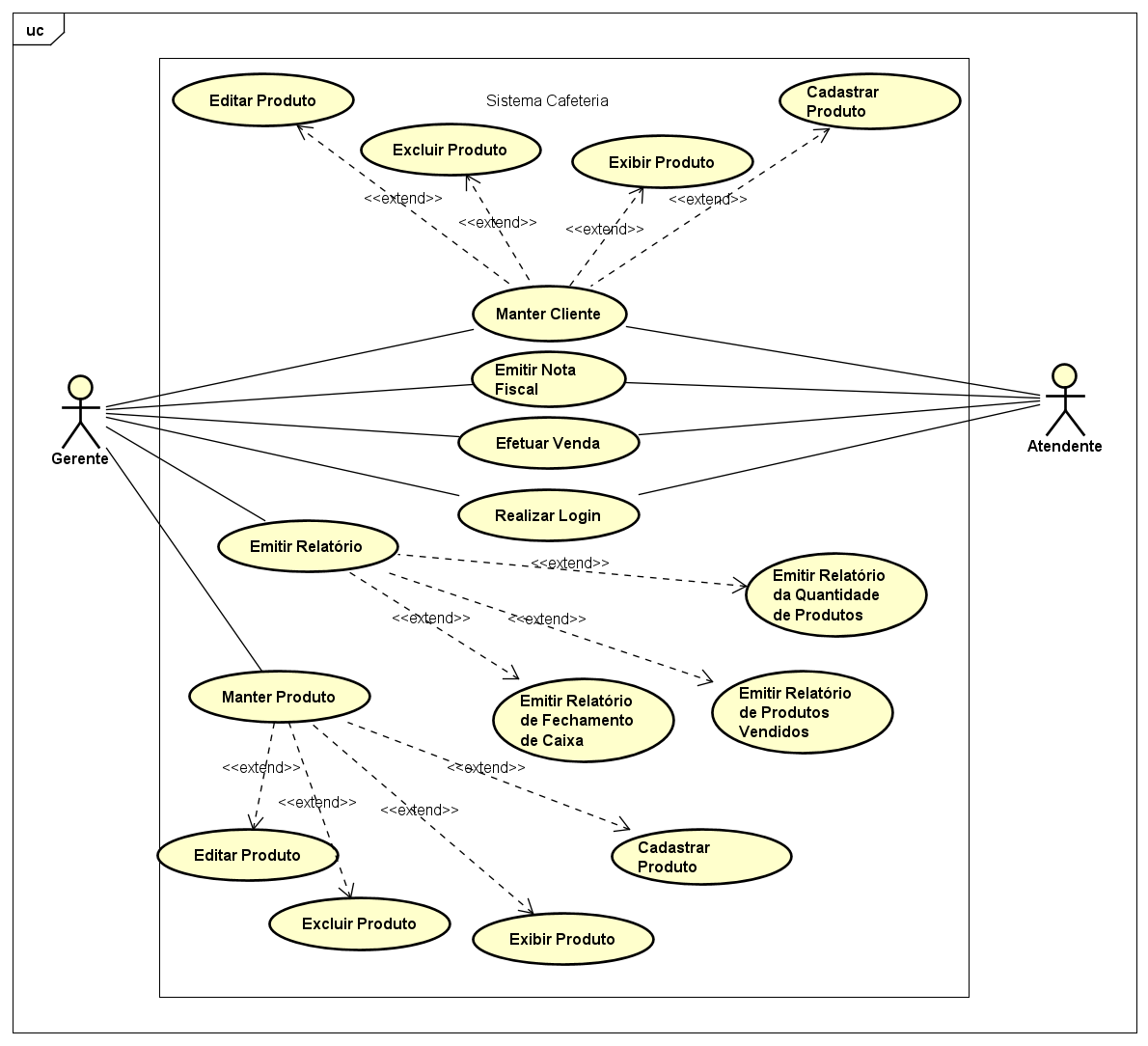
*[Descreva as visões arquiteturais usadas para descrever a arquitetura. Isto ilustra as diferentes perspectivas disponíveis para rever e documentar as decisões arquiteturais.]*

Visões Recomendadas:

* **Lógica:** Descreva a estrutura e comportamento de porções arquiteturalmente significantes do sistema. Isto deve incluir a estrutura de pacotes, interfaces críticas, importantes classes e subsistemas e as relações entre estes elementos. Isto também inclui visões físicas e lógicas dos dados persistentes.
* **Operacional:** Descreva os nós físicos do sistema e os processos, threads e componentes que rodam em cada um desses nós. Esta visão não é necessária se o sistema roda num único processo e num único thread.
* **Casos de Uso:** Uma lista ou diagrama dos casos de uso que contém requisitos arquiteturalmente relevantes.

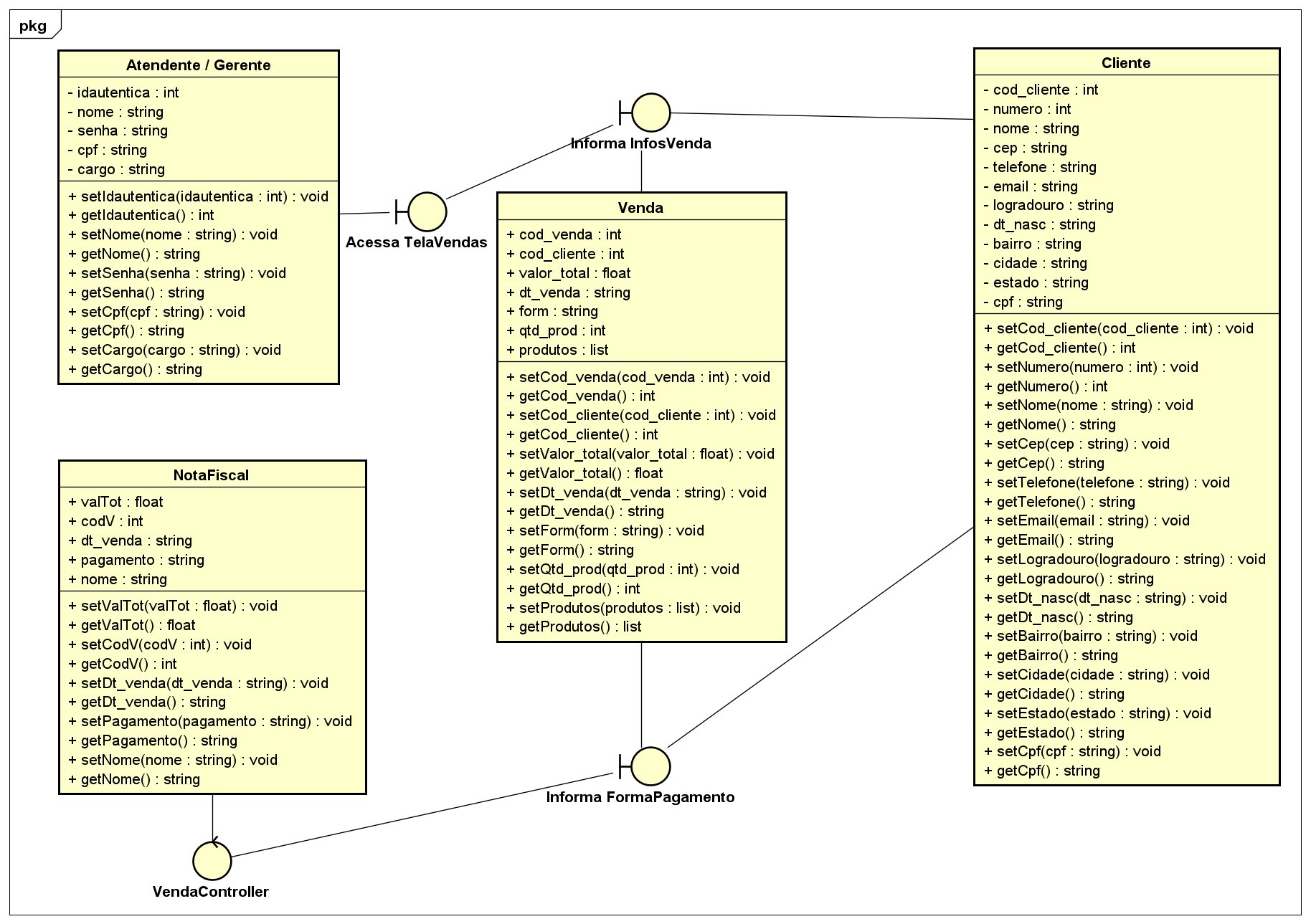
# Visão dos casos de uso

Os casos de uso do sistema Café Com Serviços serão listados abaixo.

****

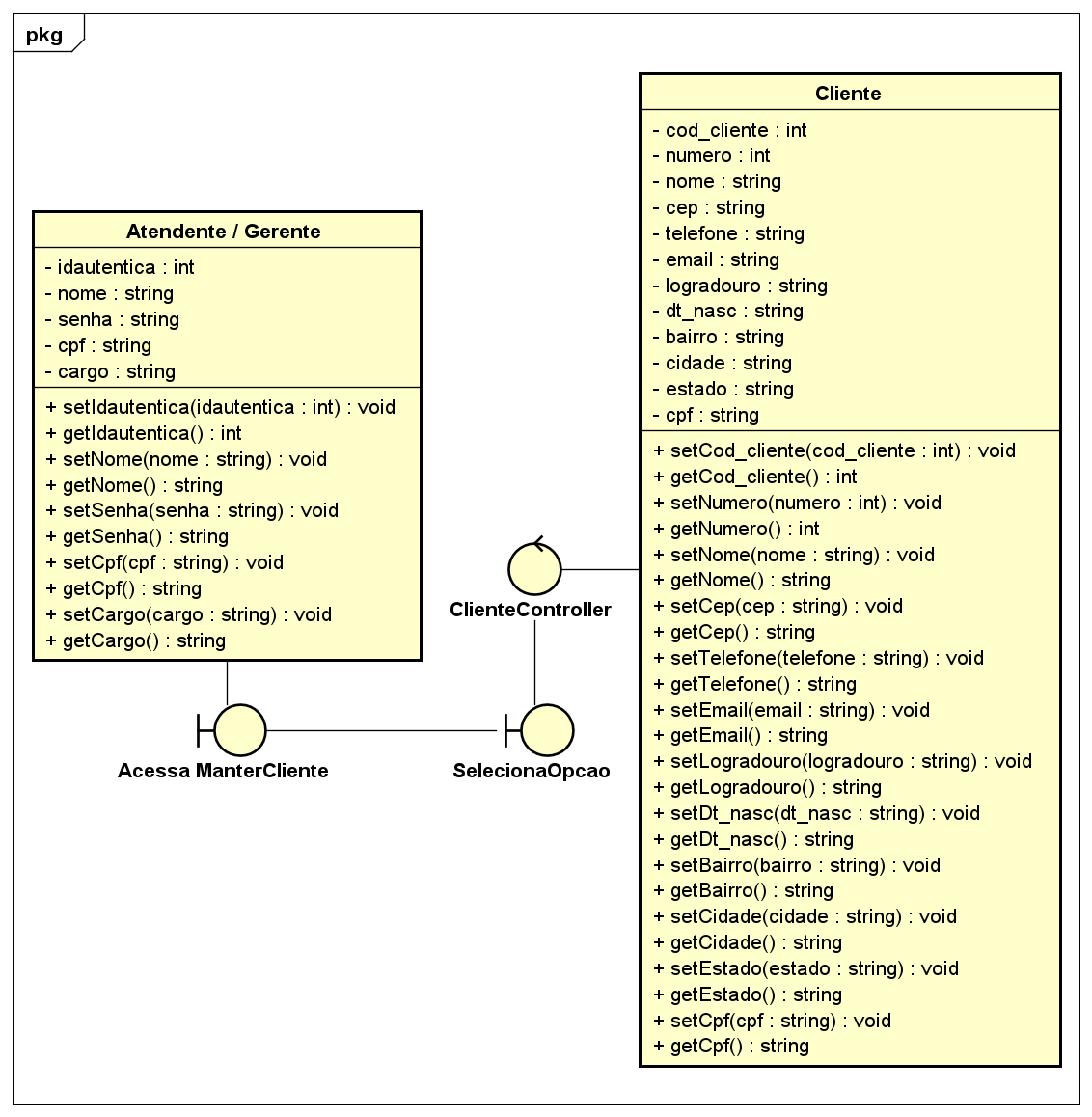
# Visões das classes participantes

# Caso de Uso Efetuar Venda

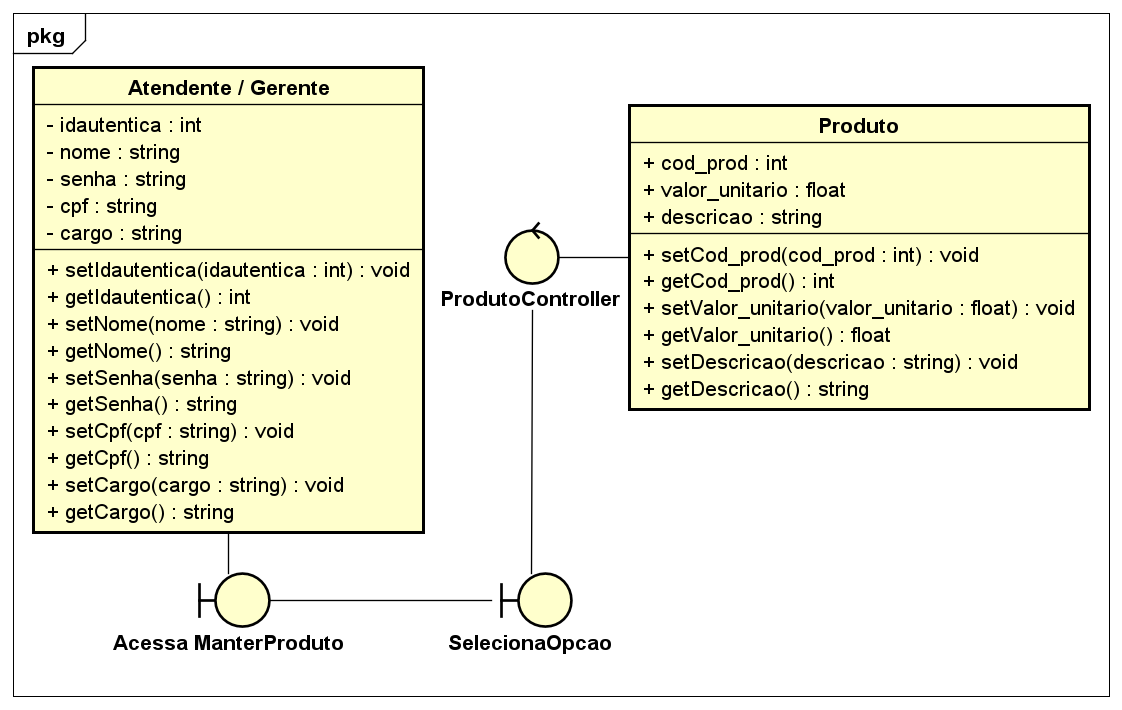


# Caso de Uso Emitir Relatório

# Caso de Uso Manter Cliente



# Caso de Uso Manter Produto



# Caso de Uso Realizar Login

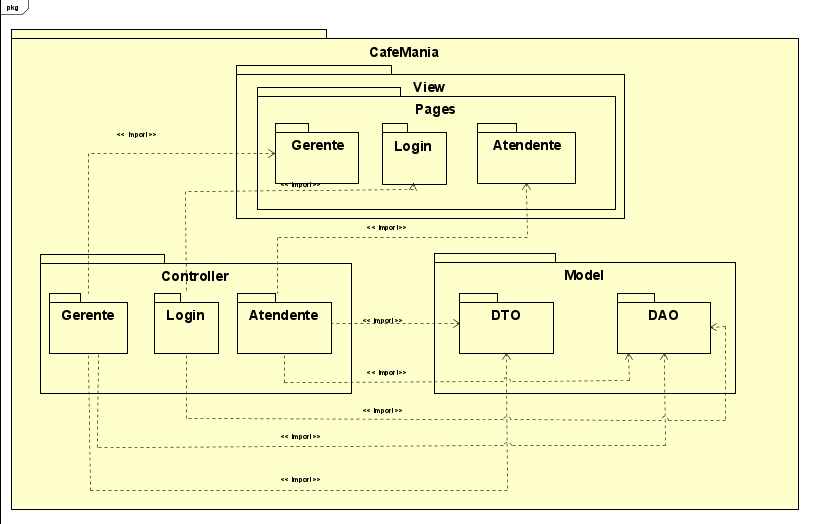
# 

# Visão Lógica

A visão lógica define a estrutura da arquitetura. Abaixo será especificado o padrão utilizado para o desenvolvimento do sistema, no caso: MVC, Microsserviços, Shared-data e Cliente-Servidor.

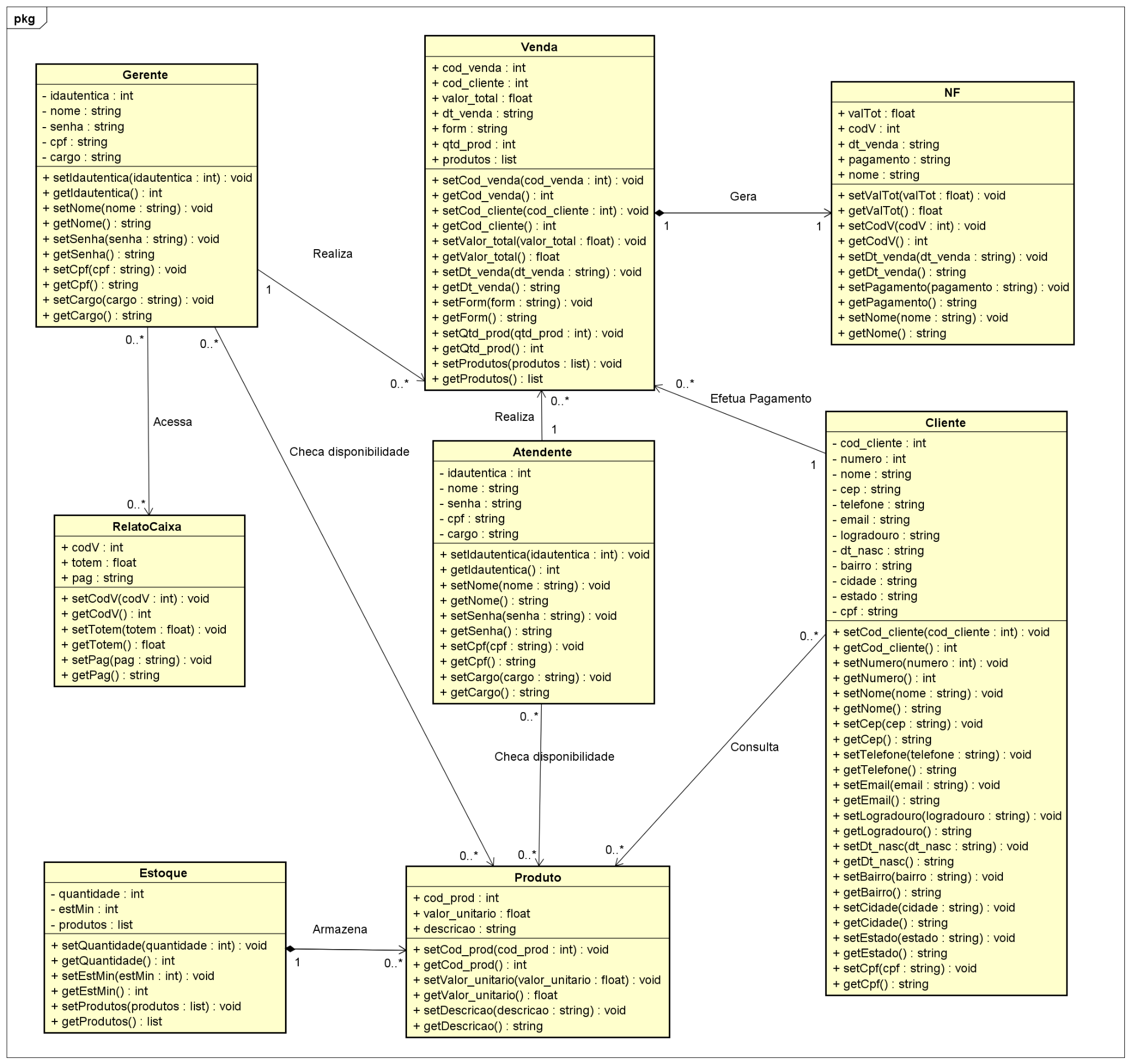
# 

# Diagrama de Pacotes da Arquitetura



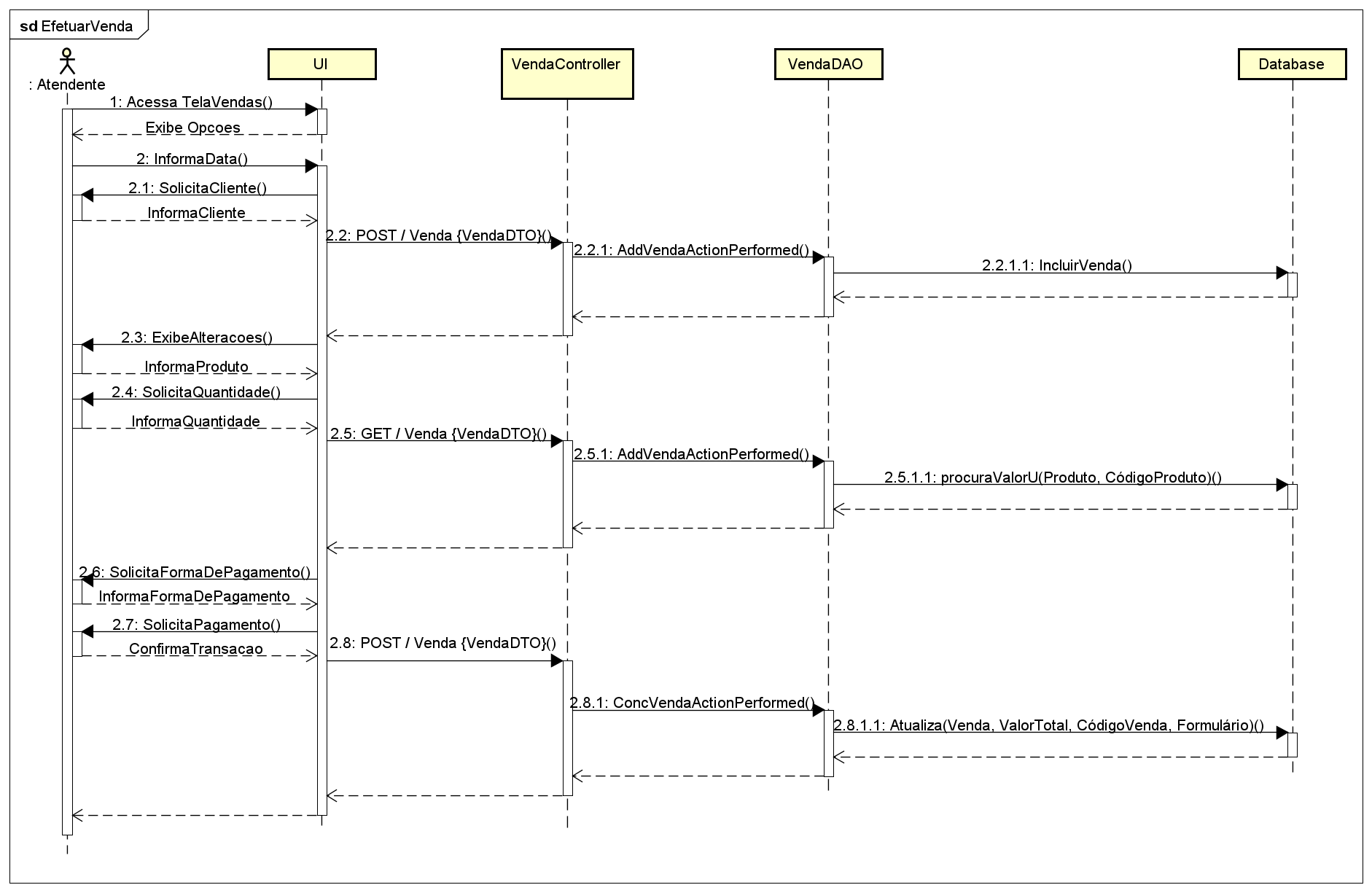
# 

# Diagrama de Classes da Arquitetura

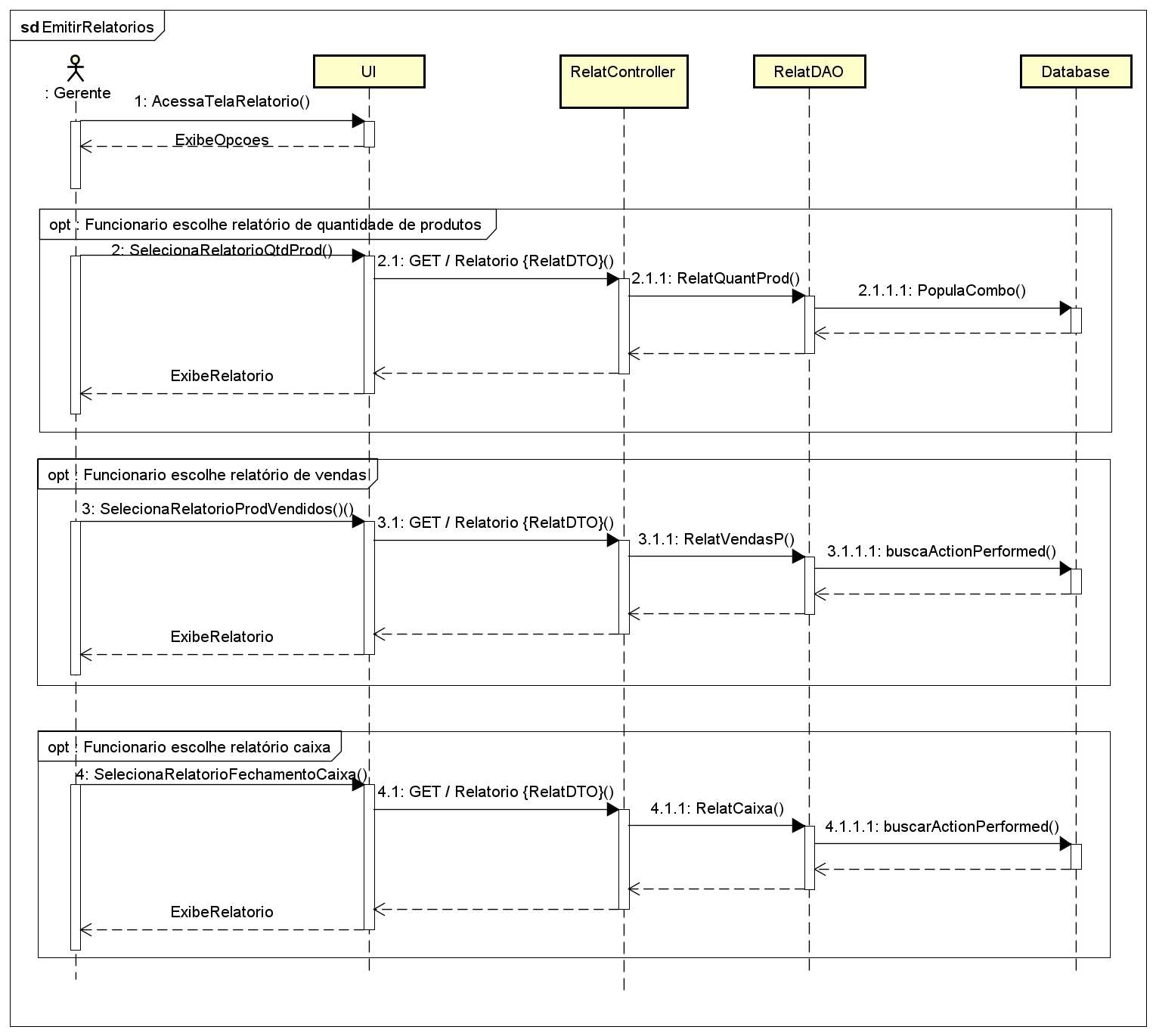


# Diagramas de Sequência

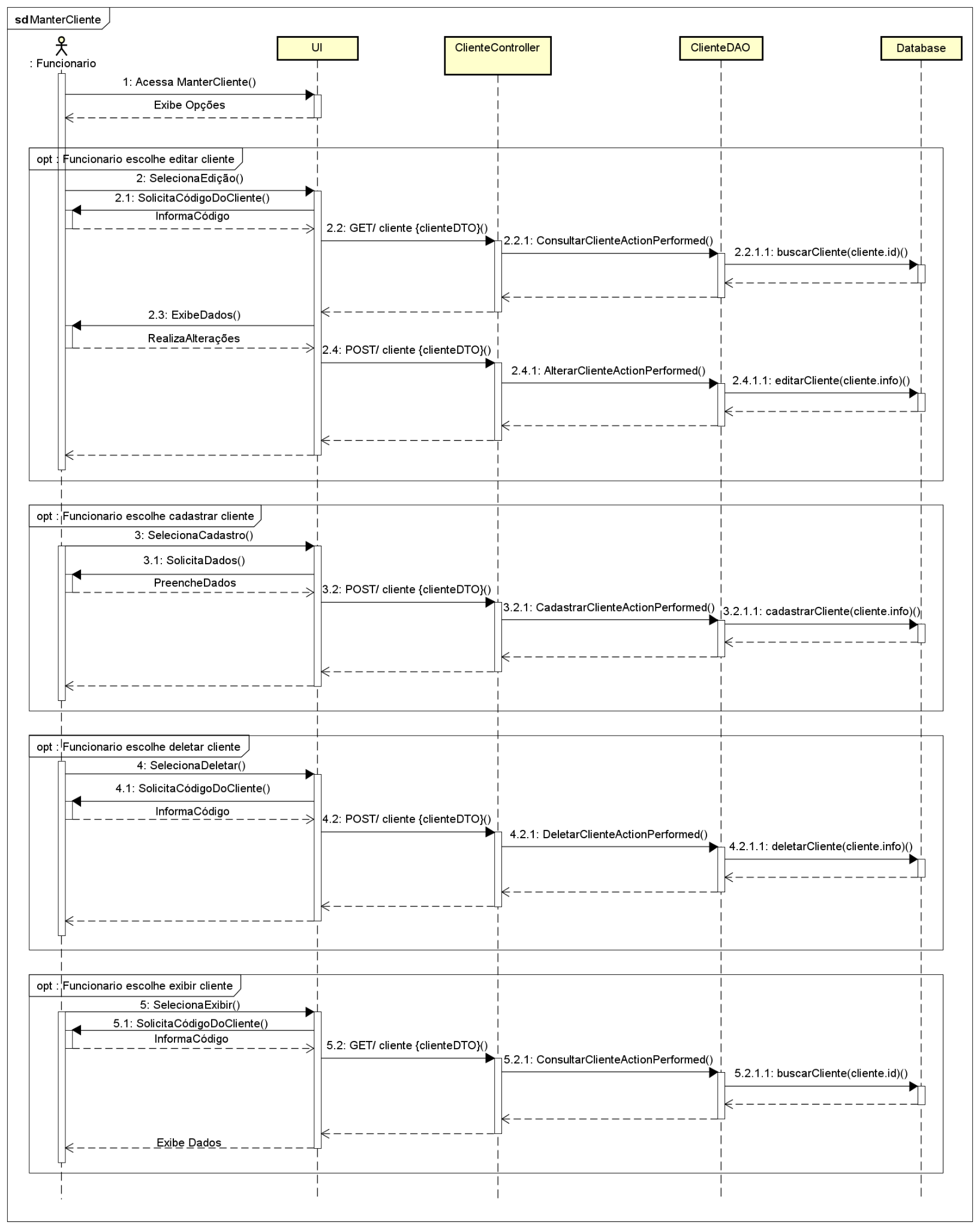
# Caso de Uso Efetuar Venda



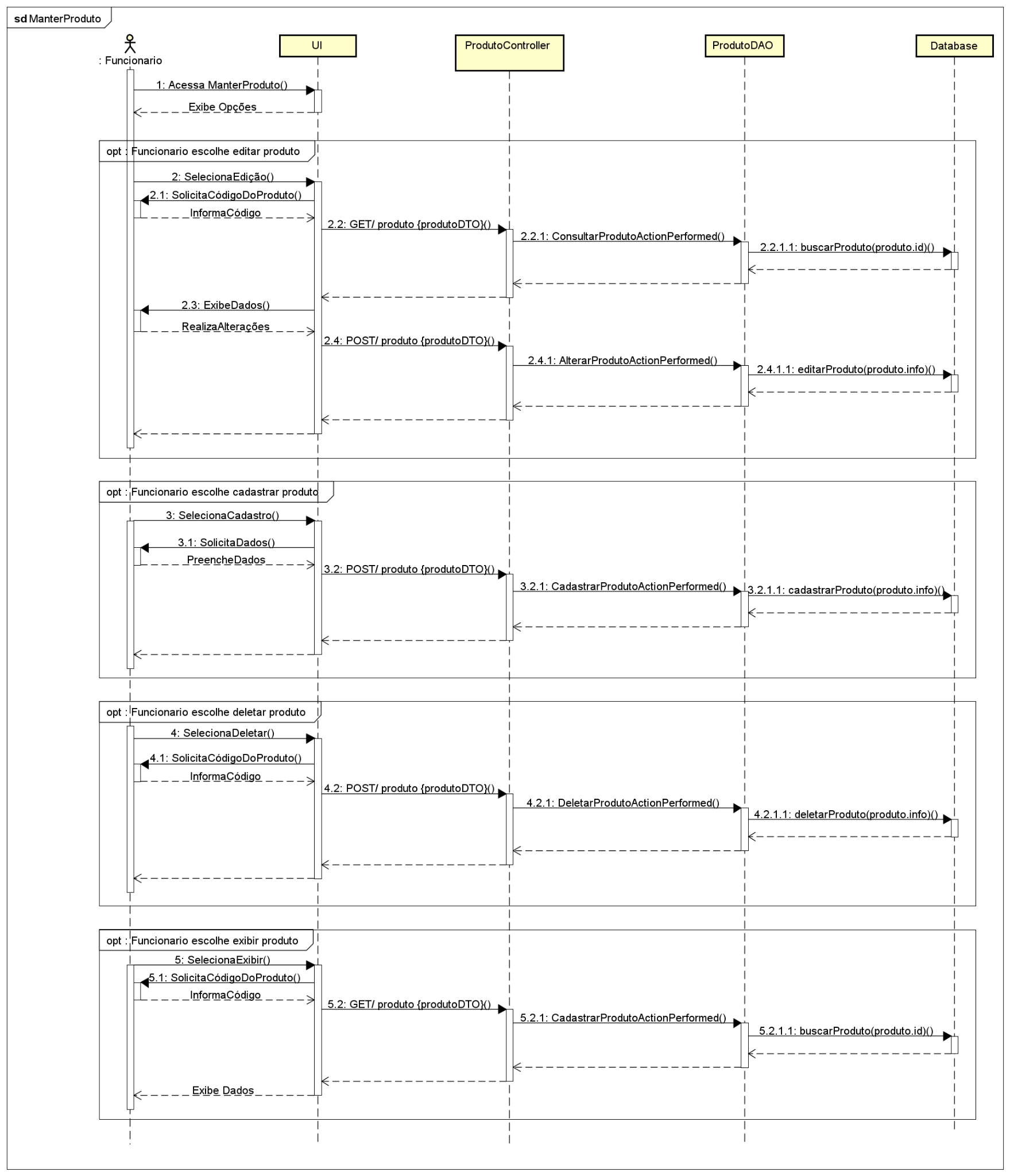
# Caso de Uso Emitir Relatório



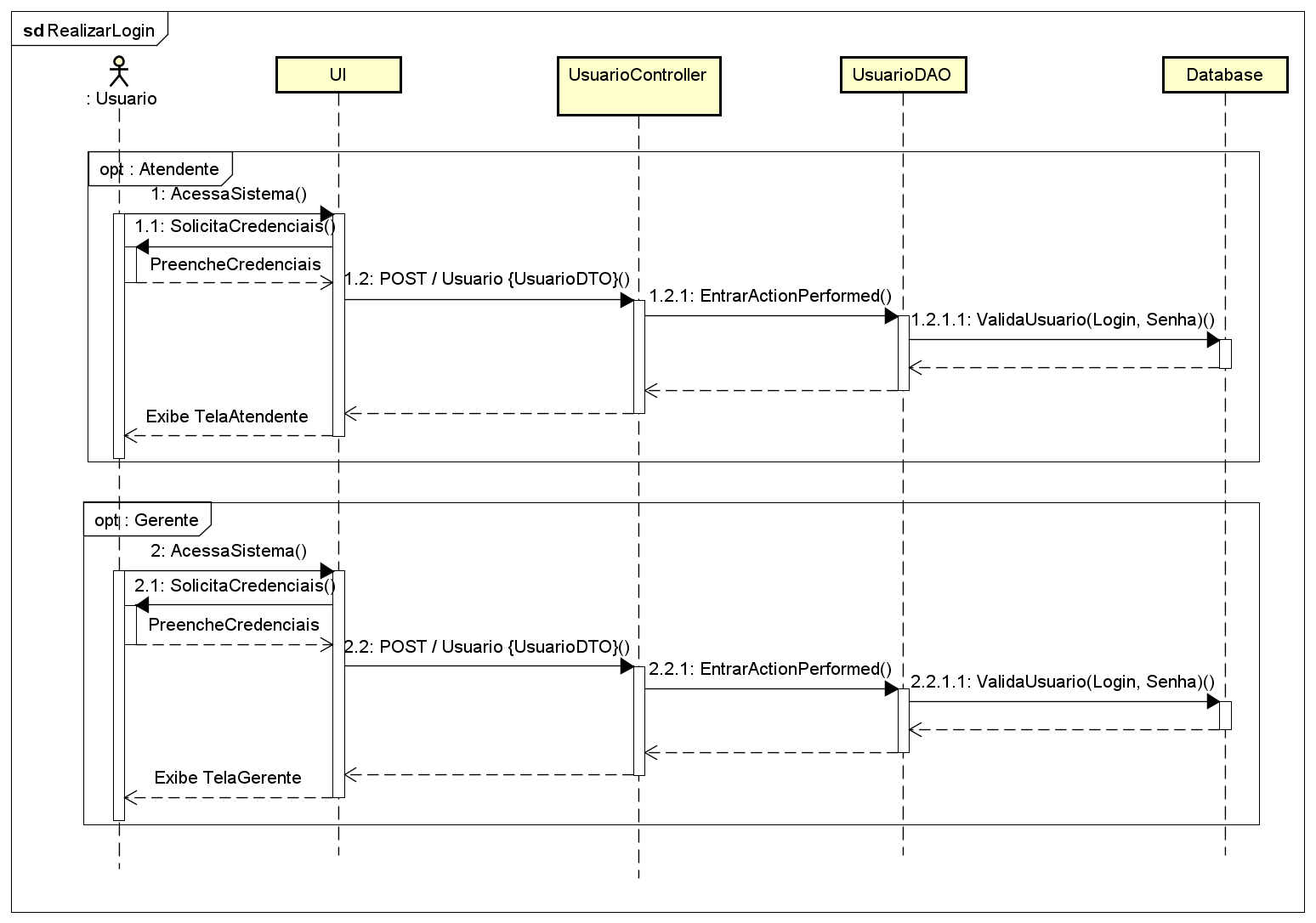
# Caso de Uso Manter Cliente



# Caso de Uso Manter Produto



# Caso de Uso Realizar Login



# Qualidade

*[Uma descrição de como a arquitetura do software contribui para todos os recursos (exceto a funcionalidade) do sistema: extensibilidade, confiabilidade, portabilidade e assim por diante. Se essas características possuírem significado especial, como implicações de segurança, garantia ou privacidade, elas deverão ser delineadas claramente.]*